The University of Jordan
Authorization Form

I, Ahmad Ramadan Hamdan, authorize the University of Jordan to supply copies of my Thesis to libraries or establishments or individuals on request, according to the University of Jordan regulations.

Signature:

Date: 25/4/2011

الجامعة الأردنيـــة
كلية الدراسات العليا

التاريــخ: / /

نموذج رقم (١٨)
اقرار والتزام بالمعايير الأخلاقية والأمانة العلمية
وقوانين الجامعة الأردنية وأنظمتها وتعليماتها
لطلبة الماجستير

أنا الطالب: أحمد رمضان ابراهيم حمدان الرقم الجامعي: (8060434)

تخصص: علم الحاسوب الكليـــــة: تكنولوجيا المعلومات

عنوان الرسالة: Digital Image Enhancement Using Cellular Automata

اعلن بأنني قد التزمت بقوانين الجامعة الأردنية وأنظمتها وتعليماتها وقراراتها السارية المفعول المتعلقة باعداد رسائل الماجستير عندما قمت شخصيا" باعداد رسالتي وذلك بما ينسجم مع الأمانة العلمية وكافة المعايير الأخلاقية المتعارف عليها في كتابة الرسائل العلمية. كما أنني أعلن بأن رسالتي هذه غير منقولة أو مستلة من رسائل أو كتب أو أبحاث أو أي منشورات علمية تم نشرها أو تخزينها في أي وسيلة اعلامية، وتأسيسا" على ما تقدم فانني أتحمل المسؤولية بأنواعها كافة فيما لو تبين غير ذلك بما فيه حق مجلس العمداء في الجامعة الأردنية بالغاء قرار منحي الدرجة العلمية التي حصلت عليها وسحب شهادة التخرج مني بعد صدورها دون أن يكون لي أي حق في التظلم أو الاعتراض أو الطعن بأي صورة كانت في القرار الصادر عن مجلس العمداء بهذا الصدد.

توقيع الطالب: .... التاريخ: ٢٤ / ٤ / ٢٠١١

DIGITAL IMAGE ENHANCEMENT USING CELLULAR AUTOMATA

By

Ahmad Ramadan Hamdan

Supervisor

Dr. Abdel latif Abu Dalhoum

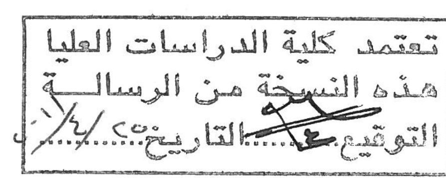
Co- Supervisor

Dr. Mohammad Qatawneh

This Thesis was Submitted in Partial Fulfilment of the Requirements for the Master's Degree of Science in Computer Science

Faculty of Graduate Studies
University of Jordan

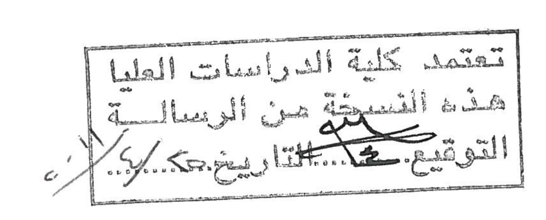
April 2011



COMMITTEE DECISION

This Thesis/Dissertation (Digital Image Enhancement Using Cellular Automata) was successfully Defended and Approved on April 10, 2011

Examination Committee	Signature
Dr. Abdel latif Abu Dalhoum, (Supervisor) Assoc. Prof. of Evolutionary Algorithms, Complex systems, and DNA Computation Models	----------------------------
Dr. Mohammad Qatawneh, (Co-Supervisor) Assoc. Prof. of Parallel Systems and Networks	----------------------------
Dr. Mohammad Alshraideh (Member) Assoc. Prof. of Software Testing Using Evolutionary Algorithms	----------------------------
Dr. Azzam Sleit (Member) Assoc. Prof. of Imaging DataBases	----------------------------
Dr. Hussein Al Ofeishat (Member) Assoc. Prof. (Al-Balqa Applied University)	----------------------------



DEDICATION

To my father and mother, and my brothers Samer and Naser; thanks for your prayers, support, and thanks for being my family.

To my beloved fiancée Aseel; thanks for your endless love, support and encouraging words.

To my whole family and all my friends, none of this would be possible without your love and support.

ACKNOWLEDGEMENT

{يَرْفَعِ اللهُ الَّذِينَ ءامَنُوا مِنكُمْ وَالَّذِينَ أُوتُوا الْعِلْمَ دَرَجَاتٍ} الزمر 9

[Allah raises the ranks of those among you who believe and those who were granted the knowledge] from the holy Quran, Surat Az-Zumar, Ayah **9**

My endless thanks to the creator, Allah, for his support and gives in my whole life and in completing this thesis.

I would like to thank my supervisors Dr. Abdel latif Abu Dalhoum and Dr. Mohammed Al-Qatawneh for their support and ideas in all stages of my work on this thesis.

Special thanks for Dr. Mohammed Al-Rawi for suggesting cellular automata as the thesis topic, and for his support and guidance. And special thanks for Dr. Bassam Hammo for his kindness and his encouragement.

TABLE OF CONTENTS

COMMITTEE DECISION..**Error! Bookmark not defined.**

DEDICATION..III

ACKNOWLEDGEMENT..IV

TABLE OF CONTENTS..V

LIST OF TABLES..VII

LIST OF FIGURES..VIII

LIST OF ABBREVIATIONS..X

ABSTRACT..XI

1. Introduction..2

1.1 Problem Definition..4

1.2 Motivation..4

1.3 Objectives..5

1.4 Thesis Organization..5

1.5 Digital Image Processing..6

1.5.1 Image Processing Operations..8

1.5.2 Image Enhancement and Restoration..9

1.5.3 Image Histogram..10

1.5.4 Image Restoration in Spatial Domain..11

1.5.5 Median Filters..12

1.5.6 Image Edge Detection..13

1.6 Cellular Automata..18

1.6.1 Building Cellular Automata..21

1.6.2 Types of Cellular Automata..25

1.6.3 Cellular Automata Classes 27

1.6.4 Examples of Cellular Automata..29

1.7 Sequential Search 30

1.8 Measurement Variables 31

2. Related Work 35

3. Research Environment and Proposed Techniques 47

3.1 Experiment Environment 47

3.2 Proposed Techniques 47

3.2.1 Iterative Weighted Edges CA 48

4. Experimental Results and Conclusions 58

4.1 Experimental Results for Salt and Pepper Noise.......................................59

4.2 Experimental Results for Uniform Noise 70

4.3 Conclusions 78

4.4 Future Work 79

REFERENCES..80

Appendix A: MATLAB Applications Source Code 86

ABSTRACT IN ARABIC ....**Error! Bookmark not defined.**

LIST OF TABLES

Number	Table Title	Page
Table (1.1)	Rule Table of Rule 30 CA	26
Table (1.2)	Rule Table of Rule 110 CA	26
Table (3.1)	Hardware specification	47

LIST OF FIGURES

Number	Figure Title	Page
Figure (1.1)	Image sampling types	7
Figure (1.2)	Common neighborhoods	9
Figure (1.3)	Histogram of 256 gray scale Lena image	11
Figure (1.4)	Gaussian filter with $\sigma = 1.4$	15
Figure (1.5)	Sobel masks to find x and y direction gradients	15
Figure (1.6)	Cellular Automata	18
Figure (1.7)	a: One dimensional CA b: Two Dimensional CA c: Hexagon CA d: One dimensional CA visualization using space-time diagram	22
Figure (1.8)	CA neighborhood	23
Figure (1.9)	CA boundaries	24
Figure (1.10)	All 8 possible neighborhoods for elementary CA Black = 1, White = 0	25
Figure (1.11)	Rule 30 space-time diagram	26
Figure (1.12)	Rule 110 space-time diagram	27
Figure (1.13)	Examples on stable (still lives) GoL patterns	29
Figure (1.14)	Examples on repeating (oscillators) GoL patterns	30
Figure (1.15)	Example on moving (spaceships) GoL patterns	30
Figure (3.1)	Neighborhood at time t contains high noise level	48
Figure (3.2)	Neighborhood at time t+1 contains low noise level	49
Figure (3.3)	Flow diagram of Iterative Weighted Edges CA	50
Figure (3.4)	Maximum HD values	54
Figure (3.5)	canny double edges result	55
Figure (3.6)	Maximum HD values	56
Figure (4.1)	Test images	59
Figure (4.2)	Results of IWE CA on Lena test image	60
Figure (4.3)	HD comparison results of applying different models on Lena test image	61
Figure (4.4)	PSNR comparison results of applying different models on Lena test image	62
Figure (4.5)	ES comparison results of applying different models on Lena test image	63
Figure (4.6)	Results of IWE CA on Wooden-boat test image	64
Figure (4.7)	HD comparison results of applying different models on Wooden-boat test image	65
Figure (4.8)	PSNR comparison results of applying different models on Wooden-boat test image	65
Figure (4.9)	ES comparison results of applying different models on Wooden-boat test image	66
Figure (4.10)	HD comparison results of applying different models on blown_ic_hr test image	67
Figure (4.11)	PSNR comparison results of applying different models on blown_ic_hr test image	67

Figure (4.12)	ES comparison results of applying different models on blown_ic_hr test image	68
Figure (4.13)	HD comparison results of applying different models on katrina_2005 test image	68
Figure (4.14)	PSNR comparison results of applying different models on katrina_2005 test image	69
Figure (4.15)	ES comparison results of applying different models on katrina_2005 test image	69
Figure (4.16)	Results of IWE CA on Lena test image	70
Figure (4.17)	HD comparison results of applying different models on Lena test image	71
Figure (4.18)	PSNR comparison results of applying different models on Lena test image	71
Figure (4.19)	ES comparison results of applying different models on Lena test image	72
Figure (4.20)	Results of IWE CA on Wooden-boat test image	73
Figure (4.21)	HD comparison results of applying different models on Wooden-boat test image	74
Figure (4.22)	PSNR comparison results of applying different models on Wooden-boat test image	74
Figure (4.23)	ES comparison results of applying different models on Wooden-boat test image	75
Figure (4.24)	HD comparison results of applying different models on blown_ic_hr test image	75
Figure (4.25)	PSNR comparison results of applying different models on blown_ic_hr test image	76
Figure (4.26)	ES comparison results of applying different models on blown_ic_hr test image	76
Figure (4.27)	HD comparison results of applying different models on katrina_2005 test image	77
Figure (4.28)	PSNR comparison results of applying different models on katrina_2005 test image	77
Figure (4.29)	ES comparison results of applying different models on katrina_2005 test image	78

LIST OF ABBREVIATIONS

Abbreviation	Meaning
CA	Cellular Automata
CCTV	Closed-circuit television
CMY	Cyan, Magenta, Yellow image coloring model
CMYK	Cyan, Magenta, Yellow, Black image coloring model
ES	Exact Similarity
GoL	Game of Life
GIS	Geographic Information Systems
HD	Hamming Distance
HSI	Hue, Saturation, Intensity image coloring model
HCI	Human Computer Interfaces
IWE	Iterative Weighted Edges
MRI	Magnetic Resonance Imaging
MSE	Mean Squared Error
PSNR	Peak Signal to Noise Ratio
PCB	Printed Circuit Board
PSM filter	Progressive Switching Median filter
RGB	Red, Green, Blue image coloring model
SFFS	Sequential Floating Forward Search
SUSAN	Smallest Univalue Segment Assimilating Nucleus edge detection method
USAN	Univalue Segment Assimilating Nucleus edge detection method

DIGITAL IMAGE ENHANCEMENT USING CELLULAR AUTOMATA

By
Ahmad Ramadan Hamdan

Supervisor
Dr. Adbel latif Abu Dalhoum

Co- Supervisor
Dr. Mohammad Qatawneh

ABSTRACT

The growing number of applications using and implementing image processing in many areas increases the demand for enhanced image processing techniques and tools. Very critical decisions may be taken on the light of image processing results; sometimes it may affect human lives if image processing is used by a medical application.

Cellular Automaton is a decentralized computational model, it is powerful, simple, and efficient technique for problem solving and modeling, despite of its simplicity, it is capable for performing complex computations with the help of only local information, these characteristics makes it successful to apply it on image processing.

Many researches were conducted to apply Cellular Automata (CA) on image processing, in this thesis we will propose a new enhanced CA model for digital image enhancement and restoration. First we will study and analyze existing CA models in literature and understanding their power and weak points, and then we propose our model and implement it, finally, we compare our results with existing results of others.

Chapter One

Introduction

1. Introduction

“We are in the midst of a revolution sparked by rapid progress in digital image processing technology” Hareish Gur.

Image processing is one of the most important and evolving areas of information technology; various kinds of applications in many fields are implementing and involving image processing, these applications are growing in all areas, this technology is a rich area of research within computer science and engineering disciplines, it is the basis for future automated visual capabilities, and it has the capability to develop machines that can perform human beings’ visual functions.

Vision is a very important information source for human beings; 75 percent of the information human being receives is visual, this importance can be concluded from the Chinese proverb “A picture speaks a thousand words”.

Today, almost all technical area are impacted in some way by digital image processing (Gonzalez and Woods, 2002), in medical field digital image processing is used in X-ray imaging, radiology, Magnetic Resonance Imaging (MRI), Ultrasound, etc. Astronomy is another field, in 1990 an incorrect mirror caused noise in telescope Hubble’s images and made them useless, but image processing techniques fixed them. Geographic Information Systems (GIS) use digital image processing for example for meteorology and terrain classification. In industry, industrial inspection is used in all kinds of industries, and Printed Circuit Board (PCB) inspection to insure that all components present and in place. Image processing techniques are used extensively by law enforcement for number plate recognition, finger print recognition, and Closed-circuit television (CCTV) video monitoring systems. Many other uses for image

processing exists, for example, Artistic effects used in movies and media, and in Human Computer Interfaces (HCI) to try to make them more natural like face recognition, and gesture recognition.

Cellular Automata (CA) became quite popular in various fields of computer graphics in the late 80s and early 90s (Gobron, et al. 2006), 2D CA are suitable for digital image processing because CA can simulate different image processing algorithms and techniques, CA have the advantage of parallel nature and constant time complexity, this makes them suitable for real time image processing (Liu, et al. 2008 and Rosin, 2010). Many researches proposed CA algorithms to perform image enhancement and restoration operations. In this thesis we will review some of these researches and we will introduce a new CA algorithm that outperforms existing techniques.

In this chapter we will define the problem which will be studied in this thesis, present thesis motivation and objectives, and then we will give an introductory background to help reader to understand the necessary terms and theories which will be discussed throw the thesis in next chapters.

This chapter will provide a brief history and a basic introduction for Digital Image Processing and Cellular Automata (CA), first we will review some image processing operations and images processing domains, discuss the concept of filtering and masking for noise removal, edge detection methods, then we will define cellular automata, give a brief history of the beginnings and evolution, describe the structure of CA, types and classes, then we will show some examples and applications of CA, after that we will explain the concept of sequential search, finally we will present measurement variables that we will use to evaluate our results.

1.1 Problem Definition

Image noise is the existence of black or white or colored dots or specks in the image, this noise affects the quality of the image, and make it difficult or even useless to extract the needed information from it (Guieb, 2007) and (Liu, et al. 2008).

Image noise occurs by many reasons; it may be because of mechanical problems in image acquisition machine used like camera or scanner (Boo, et al. 2009), or by camera lens miss focus, or because of some issue related to the environment like low or high light (Xiuqin, et al. 2008). Some kinds of noise such as salt and pepper noise may happen because of some analog to digital converter errors or bit errors in transmission. Quantization noise (uniform noise) may occur due to rounding or truncation that happens while quantizing analog image into digital image (Young, et al. 1998).

Image noise reduction is a process used to remove noise from image, and make it easier to extract its information and make it more evident and more appealing. Gaussian filtering, averaging, median filter, opening, and closing are common image noise reduction techniques (Guieb, 2007).

CA show promising results when applied to image enhancement and restoration, and we will use the concept of cellular automata to provide an alternative method for image restoration and noise removal.

1.2 Motivation

The increasing number and variety of applications using image processing causes a growing demand on image processing techniques, for some applications like medical applications, accuracy is very critical, but the existence of noise that degrades the

quality of the images makes it difficult to achieve the required level of accuracy (Guieb, 2007).

Providing new improvement on the tools and methods used to remove the noise from images will help many image processing applications to achieve better results.

1.3 Objectives

The main objective of our thesis is to propose a new CA that outperforms existing CA models, and to show that CA can outperform traditional image processing techniques. Another objective of our thesis is to empower the trend of designing special CA model for each type of noise, or generally, for each digital image processing operation.

1.4 Thesis Organization

In this chapter, we will provide a technical background for the terms and concepts which needed to be understood to proceed with the thesis, and in the next chapter we will review some of the related work which used CA in image processing to perform image enhancement and restoration.

In chapter 3, we present the methodology we used in our research, we will analyze related work techniques and try to explore their advantages and disadvantages to help us in our research, and then we will describe our proposed *Iterative Weighted Edges* (IWE) CA algorithm, which handles both uniform, and salt and pepper noise types.

Chapter 4 shows the experimental results of our proposed CA compared with related work CA, followed by conclusions and ideas for future work.

1.5 Digital Image Processing

An *image* is an artifact of some object -usually a physical object- it may be created by a special machine like camera, or handmade like a statue.

A two dimensional image can be defined by a two dimensional function *f(x,y)*, where *x* and *y* are horizontal and vertical coordinates, any pair of *x* and *y* (*x*,*y*) represent a point in the image, and the value *f* for any point (*x*,*y*) is called the *intensity (amplitude)* at that point (Gonzalez and Woods, 2002).

Digital image is a representation of image in digital computer as an array of complex or real numbers represented by a finite number of memory bits. A digital image *f(x,y)* in a two dimensional discrete space is derived from an analog image *f(x,y)* in a continuous space, where the values of *x, y* and intensity *f* are all finite and discrete (Young, et al. 1998) and (Gonzalez and Woods, 2002). All three parameters together *x*, *y* and *f* represent the basic element in an image which is called a *pixel*.

Image processing is the process of manipulating and modifying images using a wide range of techniques to achieve some required results (Efford, 2000). D*igital image processing* is the field that refers to processing digital images by digital computers. (Jain, 1989)

Newspaper industry was one of the first applications of digital image processing (Gonzalez and Woods, 2002), many other applications make use of digital image processing, such like, remote sensing via satellites and spacecrafts, medical imaging and sonar, image and video transmission for broadcasting, telecommunication, or business applications, inspection of industrial parts, radar, documentation and storage, and many other fields.

To generate a digital image from sensed data, we need to convert this continuous sensed data into a discrete digital form; this is done by applying two processes: *sampling* and *quantization*.

Sampling is the processes of digitizing the continuous coordinates of the analog image into discrete values, while *quantization* is the process of digitizing the continuous amplitude (intensity) of the analog image into discrete values.

Commonly, image is sampled to a rectangular or hexagonal finite matrix; this matrix is easily represented and stored in digital computer memory.

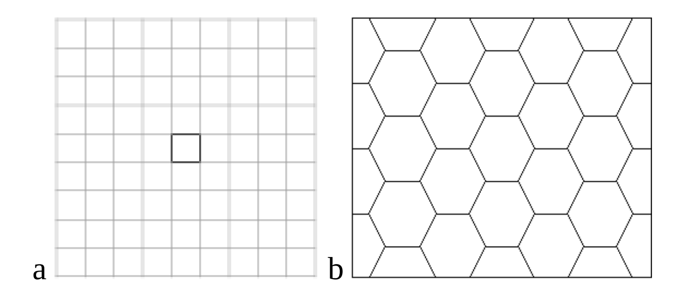


Figure (1.1) Image sampling types **a:** rectangular sampling **b:** hexagonal sampling

Images be quantized into two possible values for each pixel, 1 for white or 0 for black, in this case the image is called a binary image, and only one bit is required to store each pixel of the image in a digital computer, but if the image is quantized into one of 256 possible values, a gray scale image results, 8 bits will be required to store each pixel of the image.

For colored images, many color models are used to facilitate the specification of colors and represent them, most common color models are RGB (red, green, blue) which is used for color monitors and color video cameras, CMY (cyan, magenta,

yellow) and CMYK (cyan, magenta, yellow, black) which are used for printing, and HSI (hue, saturation, intensity) which is very close to the way human interprets colors.

RGB model is an additive model, where the color is generated by adding the values of red, green, and blue, an RGB image consists of three matrices, one for red, one for green, and one for blue, these matrices are placed on top of each other in a 3D representation, RGB image can be given by the function f(x, y, R, G, B) where the mix of R, G, and B values gives one of 2^{24} different colors.

1.5.1 Image Processing Operations

Operations that can be applied on digital images to perform image processing on them can be classified into three categories:

1. Point: The result of processing a specific coordinate depends only on its value.
2. Local: The result of processing a specific coordinate depends on the values of its neighborhood.

   Neighborhood is the finite set of pixels around the pixel under processing, defining the neighborhood is an important role in modern digital image processing, most common neighborhoods are 4-connected (well known as Von Neumann neighborhood), and 8-connected neighborhood (well known as Moore neighborhood) for rectangular sampling, and 6-connected neighborhoods for hexagonal sampling.

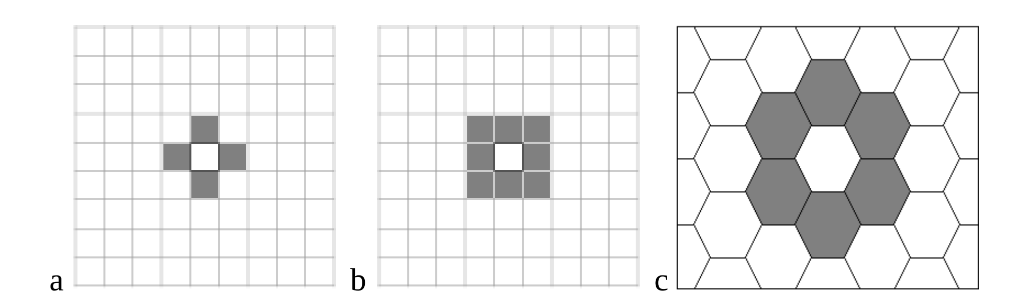


Figure (1.2) Common neighborhoods **a:** 4-connected neighborhood **b:** 8-connected neighborhood **c:** 6-connected hexagon neighborhood

3. Global: The result of processing a specific coordinate depends on the values of all coordinates in the input image.

1.5.2 Image Enhancement and Restoration

Image enhancement is the process of producing a more pleasing image from a captured or stored original image by improving its quality. Images are enhanced by executing various mathematical operations using digital computers to improve their appearance and make them more useful and pleasing to human viewers or to use them in other image processing systems.

Image enhancement operations can be simple like making an image brighter or darker, increase or decrease its contrast, or to resize or crop a required part of an image, and it can be more complex as enhancing images for medical or industrial applications.

Image restoration is the process of producing the best estimate for an original image starting from a degraded image, degradation may happen due to some mechanical problems during image acquisition, or due to some noise, motion blur, or miss focus.

Image Noise affects the quality of the image, and makes it difficult or even useless

to extract the needed information from it. Image noise reduction is an image restoration process used to remove noise from image, and make it easier to extract its information and make it more evident and more appealing. Gaussian filtering, averaging, median filter, opening, and closing are common image noise reduction techniques. (Guieb, 2007)

Image enhancement and image restoration fall into two categories: spatial domain and frequency domain methods. Spatial domain approaches interact directly with image pixels, while frequency domain approaches deals with the Fourier transform of the image (Gonzalez and Woods, 2002). Fourier transform is a mathematical operation that decomposes signals of the image into their constituent frequencies.

1.5.3 Image Histogram

Image histogram is a presentation of image intensity distribution across the full range of intensity values. Formally, for a gray level image, a histogram is a discrete function $h(r_k)=n_k$ where r_k is the kth gray level (intensity) and n_k is the number of image pixels having gray level n_k. (Gonzalez and Woods 2002)

Histogram gives some information about the image, such like, its brightness, darkness, and its contrast, Figure (1.3) shows histogram of the standard test image Lena with 256 gray-scale.

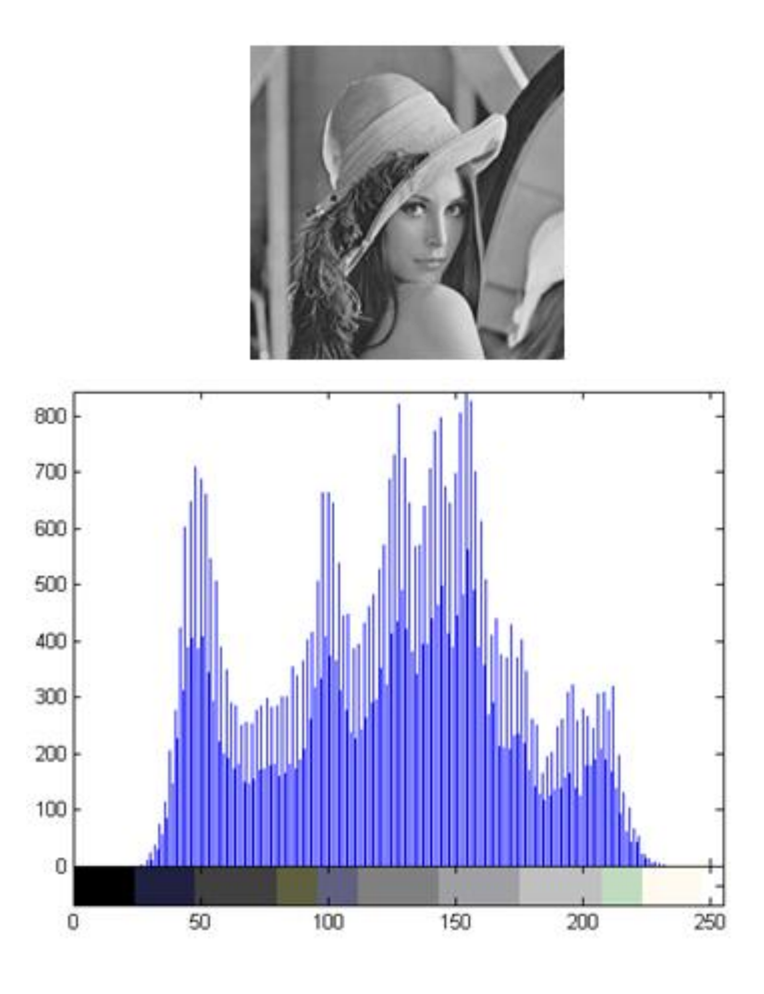


Figure (1.3) Histogram of 256 gray scale Lena image

1.5.4 Image Restoration in Spatial Domain

This section will give an introduction for some techniques used for image restoration which will be mentioned later in the thesis, first we will discuss the concept of image processing in spatial domain, define masks (filters), then we discuss some techniques that use filters to perform image restoration, these techniques are median filter, center weighted median filter, switching median filter, adaptive median filter, and Progressive Switching Median (PSM) filter.

Any process done on the image in special domain can be denoted by the expression:

$$g(x,y) = T[f(x,y)] \quad \ldots (1.1)$$

Where $f(x,y)$ is input image, $g(x,y)$ is output image, and T is transition function applied on each pixel and defined over some *neighborhood*.

One of principal approaches for transition functions is to use *masks* (also called *filters*), a mask is a small 2D array (commonly 3x3), this array has values, these values determine the nature of process applied on the input image $f(x,y)$. Image processing using masks is often referred to as *mask processing* or *filtering*

1.5.5 Median Filters

Standard median filter is the earliest median filter introduced, it uses a mask that is applied on each pixel in the input image. The median value of the pixels under the mask is evaluated then replaced by the value of the pixel in the center of the mask.

Median filter is both simple and effective to remove noise from images, it was used for long time, however, its performance depends on the size of the mask used, and it has a well known side effect of detail depression. (Xiuqin, et al. 2008) and (Chang, et al. 2008)

Center weighted median filter was introduced to solve the limitation of standard median filter, in this filter the center pixel is repeated n times before evaluating the median value of the pixels under the mask, this filter sacrifices of noise removing capability to preserver image details.

Switching median filter combines median filter with impulse detector (Wang and

Zhang, 1999) and (Zhang and Karim, 2002), the impulse detector tries to classify the pixel under processing whether noised or not, if classified as noised, median filter is applied, otherwise, the pixel left untouched. (Ghanekar, et al. 2007)

Adaptive median filter uses a complicated impulse detection scheme; it classifies each pixel in one of four types: uncorrupted, isolated impulse, non-isolated impulse, and edge pixel, after the classification is done, standard median filter is applied on isolated and non-isolated impulse pixels, and fuzzy weighted median filter is applied on other noised pixels. (Chang, et al. 2008 referring Eng and Ma, 2000 and Eng and Ma 2001)

Progressive Switching Median (PSM) filter was introduced by Zhang and Zhang in 1999 to avoid affecting not noised pixels, the algorithm consists of two main steps, the first, is an impulse detection algorithms that detects noise affected pixels (called switching scheme), the second step runs both impulse detection algorithm and noise filtering for several number of iterations in this phase the noise affected pixel is replaced with an estimated value calculated only from not noised pixels in the neighborhood (Wang and Zhang, 1999) and (Kulkarni, et al. 2010) and (Boo, et al. 2009)

1.5.6 Image Edge Detection

An *edge* is an area in the image where there is a jump of intensity from one pixel to another adjacent pixel, in other words, it’s a gathering area of pixels which have strong intensity changes (Zhang, et al. 2008); this causes the edge area to have a strong intensity contrast.

Edges are very important in many research areas; they contain useful information

for identifying and extracting objects from images, edge detection received much attention in the last two decades, it is very important preprocessing step for many applications; because it prepares the image for object segmentation and feature extraction used in pattern recognition. (Kumar and Sahoo, 2010) and (Neoh and Hazanchuk, 2005)

1.5.6.1 Canny Edge Detector

In 1983, John Canny created an edge detection approach for his master thesis (Nadernejad, 2008 referring Owens, 1997), and in 1986 he published “A Computational Approach for Image Edge Detection” where he introduced his approach that was known later as Canny Edge Detector, this approach is widely considered as a standard for image edge detection (Nadernejad, 2008), Canny edge detector is a multi-stage algorithm to detect edges in images.

Canny put three performance criteria for his algorithm (Canny, 1986):

1. Good detection: The algorithm should mark as many real edges as possible with low probability of marking non-edge pixels.
2. Good localization: The marked edges should be as close as possible to real edges.
3. One response to a single edge: If there is more than one response to a single edge, then only one must be marked as an edge, and the others should be marked false.

The algorithm runs in 6 steps as follows:

1. The input image is first filtered to remove any unwanted noise by Gaussian

filter, for example with $\sigma = 1.4$

1/115

2	4	5	4	2
4	9	12	9	4
5	12	15	12	5
4	9	12	9	4
2	4	5	4	2

Figure (1.4): Gaussian filter with $\sigma = 1.4$

2. Find the edge strength by finding the gradient operator, gradient can be estimated using any gradient operator like Roberts, Sobel, Prewitt, etc. For example, Sobel operator uses two 3x3 masks one for x direction, and the other for y direction

+1	+2	+1
0	0	0
-1	-2	-1

Gx

-1	0	+1
-2	0	+2
-1	0	+1

Gy

Figure (1.5): Sobel masks to find x and y direction gradients

Then the edge gradient is calculated using the formula $|G| = |Gx| + |Gy|$, then Find the direction of the edge using the formula $\theta = \tan^{-1}(Gy/Gx)$

3. Estimate the direction of the edge to the nearest direction of the following: 0 degrees, 45 degrees, 90 degrees, or 135 degrees.
4. Apply non-maximum suppression by tracing the edge by its direction and suppress (set to 0) the pixels that is not considered as edge, this gives a thin edge as a result.
5. Use two thresholds T_1 and T_2, where $T_1 < T_2$, to threshold the result of step 4,

the result of applying T_2 has less noise and fewer false edges, but it has larger gaps between edges (edges appear as dashes).

6. Link “dashed” edge segments of T_2 by tracing each segment till its end, then search the neighbor edges of T_1 to fill the gap between current segment and next segment it T_2.

Canny edge detector still considered a state of the art edge detector (Shapiro and Stockman, 2001), and it is hard to find an edge detector that is better than Canny’s.

1.5.6.2 SUSAN Edge Detection Method

SUSAN stands for Smallest Univalue Segment Assimilating Nucleus, a new approach for edge and corner detection with noise suppression. It was introduced in 1997 by Stephen M. Smith, J. Michael Brady.

SUSAN method four criteria for edge detection which are:

1. Good detection: The algorithm should mark as many real edges as possible with low probability of marking non-edge pixels.
2. Good localization: The marked edges should be as close as possible to real edges.
3. Only one response to a single edge: If there is more than one response to a single edge, then only one must be marked as an edge, and the others should be marked false.
4. Speed: Algorithm performance should be good as it will be useful for real time image processing applications.

SUSAN edge detector uses circular masks, the common radius of the mask is 3.4 pixels, this gives a mask with 37 pixels, however, a 3x3 mask can be used too. (Smith and Brady, 1997)

The algorithm works by placing the mask on each pixel in the processed image, then the following three steps are executed: (Smith and Brady, 1997)

1. Compare every pixel in the neighborhood $\vec{m} \in M$, where ***M*** is the neighborhood, with the center pixel $\vec{m}_0$ using the following formula

$$c(\vec{m}) = e^{-\frac{(I(\vec{m}) - I(\vec{m}_0))^6}{t}} \quad \ldots (1.2)$$

   Where *c* is the output of the comparison, *I* is the brightness value of the pixel (Intensity), *t* is a threshold. The power 6 was used as it is shown to be theoretical optimum.

2. Optain the total of all comparisons of all image pixels obtained in step 1 (*c*)

$$n(M) = \sum_{\vec{m} \in M} c(\vec{m}) \quad \ldots (1.3)$$

3. Now, the initial edge response is created by comparing the value of *n* with the fixed threshold $g=3n_{max}/4$, called *geometric threshold*, where n_{max} is the maximum value *n* can have, using the following rule:

$$R(M) = \begin{cases} g - n(M) & \text{if } n(M) < g \\ 0 & \text{otherwise,} \end{cases} \quad \ldots (1.4)$$

If the image is not noised, then there is no need to execute this third step.

1.6 Cellular Automata

A Cellular Automata (CA) is a decentralized computational model that performs complex processing with the help of local information, this model is made up of a number of individual components (cells) each of those cells contains an automaton. The communication between cells is limited to local interaction, and each cell in a specific state changes its state over time according to the states of its local neighbors "local rule", a complex behavior emerges from these simple rules, this complex behavior is what makes CA interesting. (Ganguly, et al. 2001) and (Niesche 2006)

Cellular automata describe complex system by interaction of simple individuals following simple rules instead of describing them with complex equations.

Cellular Automata (CA) are simple dynamic discrete models that perform complex behavior; it can be imagined as an array of identical programmed automata that interact with each other, Figure (1.6). (Ye and Le, 2008)

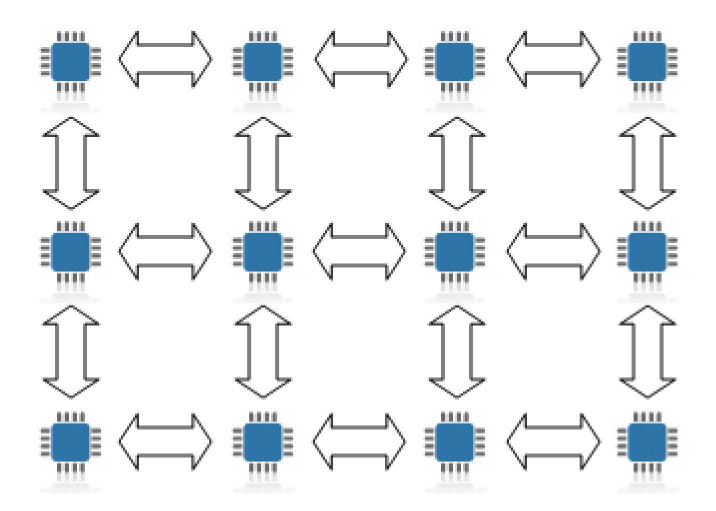


Figure (1.6) Cellular Automata

Cellular Automata can be defined by the four-tuple CA = {S, L, V, T}, where S is a regular grid of cells, each can be in one of *k* possible states belonging to a finite

alphabet L, S_i is the state of each cell at a given time, T is the transition function, the neighborhood V is defined by the number v and location of the neighboring cells considered by the transition function (Guieb, 2007).

Both mathematicians Stanislaw Ulam and John von Neumann worked together in the nineteen forties at the Los Alamos National Laboratory located in Los Alamos, New Mexico. Stanislaw Ulam was studying growth of crystals, he used the lattice network by Enrico Fermi as a model for his study, he was trying to generate graphical constructions using specific rules and he noticed that simple rules were generating complex and sometimes repetitive results. (Gauci, 2008)

John Von Neumann in 1940s who was working on the problem of self-replicating systems, he was trying to develop a self replicating machine. The model he used first, which is kinematic model, was too complex and needed too rapidly increased cost (Gauci, 2008), Neumann's colleague Stanislaw Ulam suggested that Neumann must use a mathematical abstraction to develop his design around, like the one Ulam used in studying growth of crystals, this freed Neumann's experiments from physical world and made him to start working on simplified universe, thus the first cellular automata was born.

Von Neumann's cellular automata are two-dimensional, and the self-replicator implemented algorithmically, the design was known as the tessellation model and called Von Neumann's universal constructor.

Norbert Wiener and Arturo Rosenblueth developed a model of excitable media using cellular automaton in 1940s, their model was motivated by the mathematical description of impulse conduction in cardiac systems, modern research publications on

cardiac arrhythmia and excitable systems still refer to their work.

In 1960s, and for the first time, a connection between cellular automata and mathematical field of dynamical systems was established, G. A. Hedlund’s paper in this field still considered as a seminal paper for mathematical study of cellular automata

The Game of Life (GoL) cellular automata model invented by John Conway and popularized by Martin Gardner became very widely known In 1970s. It is a two state, two dimensional cellular automata model, Each cell has two possible states, ”Alive” and ”Dead”, represented by drawing the corresponding square black or white, respectively, GoL rules are: If the cell is alive then it stays alive (survives) if it has two or three live neighbors, otherwise it dies of loneliness or overcrowding. If the cell is dead then it becomes alive if it has exactly three living neighbors.

Game of life achieves very impressive diversity of behavior despite of its simplicity; this behavior fluctuates from apparent randomness and order, the gliders, which are arrangements of cells that move themselves across the grid, gliders frequent occurrence is one of the most important features of GoL, automaton can be arranged in a way that gliders interact to perform computations.

In 1969, Konrad Zuse published his book Calculating Space (Zuse, 1969), this book was the first book on what we call today digital physics, in his book he showed that the universe is the output of a deterministic computation on a giant cellular automaton.

In 1983 Stephen Wolfram published a series of papers about elementary cellular automata, and in 2002 he published his book A New Kind of Science which is today one of the most cellular automata references.

1.6.1 Building Cellular Automata

1. Cell and cell state

The cell is the basic element of CA, it represents a memory slot that holds a *state*, a state can be one of a finite set of states, in the simplest scenario, state space is binary in which the state can be 0 or 1, but in more complex cases, cell sate can be one of more than two states up to finite number of states.

2. Lattice

Lattice is the arrangement of cells, cells can be arranges in one straight line, in this case CA are called one dimensional, most common CA are two dimensional CA, CA can have hexagonal shape. Visualization of 1D CA is very easy, but it is difficult for CA with higher dimensions, for 1D CA, evolution is visualized using space-time diagram where consecutive configurations are drawn under each other. Horizontal lines represent space, and time increases downwards, as shown in Figure (1.7) d. Complexity of CA increases as the number of dimensions increases.

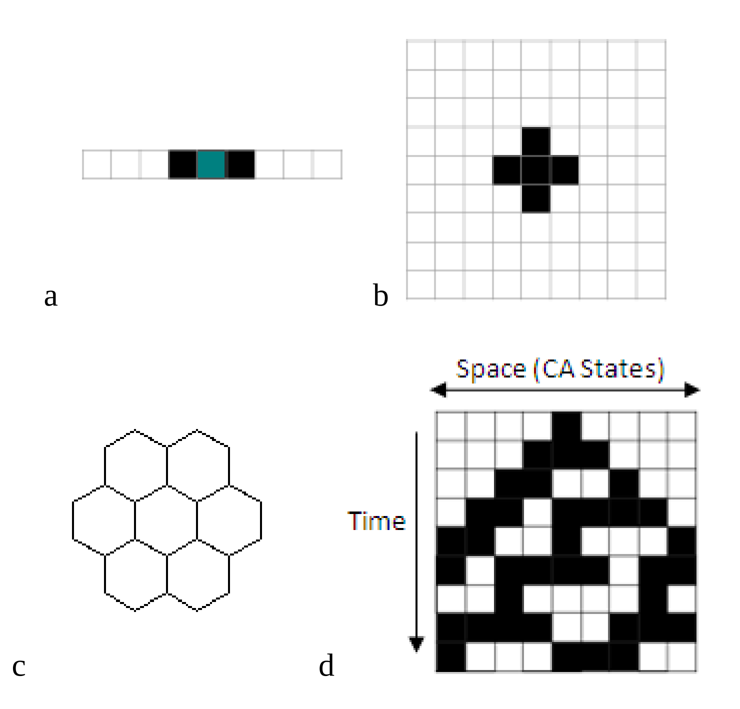


Figure (1.7) a: One dimensional CA **b:** Two Dimensional CA **c:** Hexagon CA **d:** One dimensional CA visualization using space-time diagram

3. Neighborhood

Neighborhood is the finite set of cells surrounding a cell and that influences its next state. The choice of neighborhood influences the behavior of the cellular space. Considering two dimensional CA, the common neighborhood definitions are:

Von Neumann Neighborhood: Four cells (or five if center cell is considered in neighborhood), this neighborhood includes the north, south, east, and west cells, the default radius is r=1, as shown in Figure (1.8) a, but it can be extended to a larger radius, Figure (1.8) b shows extended Von Neumann neighborhood with radius r=2

Moore Neighborhood: this neighborhood is an enlargement of Von Neumann neighborhood; it contains also diagonal cells (north-east, north-west, south-east, and south-west cells), Moore neighborhood can also be simple with radius r=1 or extended

with radius r>1 Figure (1.8) c and d shows Moore neighborhood with radius r=1, and r=2 respectively.

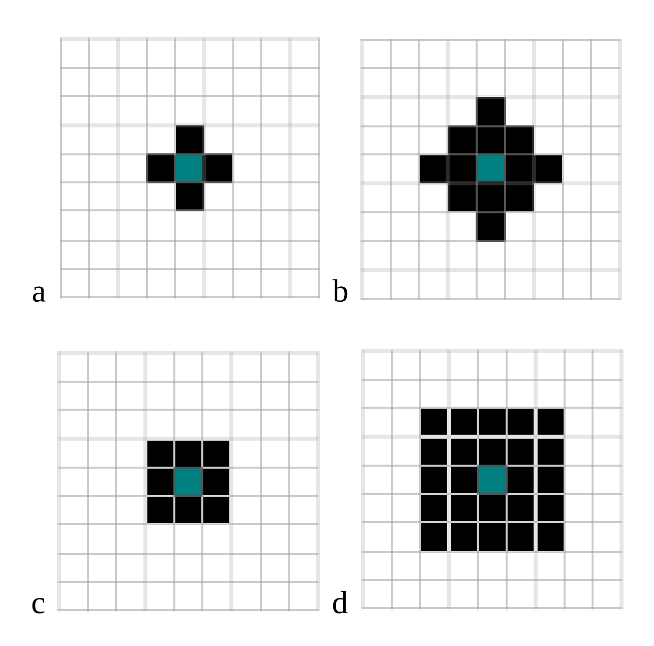


Figure (1.8) CA neighborhood **a:** Von Neumann neighborhood **b:** Extended Von Neumann neighborhood **c:** Moore neighborhood **d:** Extended Moore neighborhood

4. Transition function

Transition function T (also called rule) is the rule that gives the next state of automaton central cell (at time t+1) as a function of its own and its neighbors' state, it may be an algorithm or a simple calculation, the rule is applied on each cell in cellular automata at time t to get a new generation at time t+1

5. Boundaries

For the cells which are in the inner lattice, the neighborhood fits with existing lattice cells, but for cells on the boundaries some cells which are supposed to participate in the neighborhood des not exist. There are number of approaches to deal with cells on

the boundaries and there is no restriction for the cellular automaton located on the boundaries of the lattice.

One approach is to consider cells outside the lattice as null (blank), another is to duplicate the cells at the boundaries, a third approach is to wrap-around the boundaries as if the cells are cyclic, a fourth approach is mirrored boundary, where for example to create a mirrored boundary for a left most cell in the lattice, its right neighbor is mirrored to be a left neighbor (Selvapeter and Hordijk, 2009)

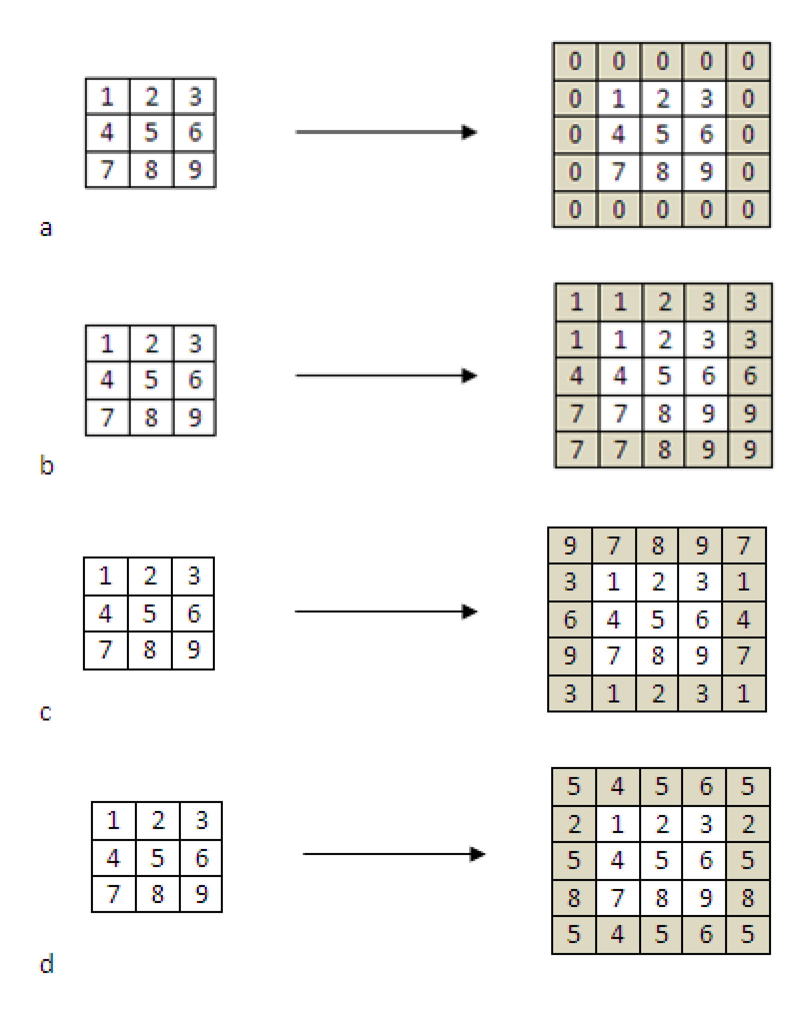


Figure (1.9) CA boundaries **a:** Null boundaries **b:** Duplication boundaries **c:** Reflective boundaries **d:** Mirrored boundaries

6. Number of generations

Applying transition function can be repeated for a specific number of times

according to the problem, or until CA become stable and no more change in its statuses takes place.

1.6.2 Types of Cellular Automata

1. Elementary Cellular Automata

Elementary cellular automata is the simplest CA; it is one dimensional CA with two possible states per cell, 0 or 1 (binary CA), the neighborhood consists of the cell itself and its adjacent right and left cells, this means that there are 8 possible configurations for neighborhood as shown in Figure (1.10).

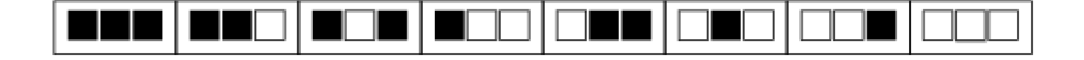


Figure (1.10) All 8 possible neighborhoods for elementary CA
Black = 1, White = 0

Transition function (rule) decides for each configuration of neighborhood what will be the next state for center cell (will it be 0 or 1), this means that there will be 2^8=256 rules, in general, the number of possible rules for a generalized *d* dimensional CA with *S* states and a neighborhood of *n* levels (also called radius, usually 1), is given by $R=S^{S^{(2n+1)^d}}$ (Wolfram, 2002)

Wolfram notation is used to refer these rules, Wolfram notation is a standard naming convention in which the name of CA rule is the decimal number which binary representation gives the rule table, with eight possible neighborhoods are listed in reverse counting order.

For example rule table of Rule 30 CA (30 in binary is 11110) is given below:

Table (1.1): Rule Table of Rule 30 CA

current pattern	111	110	101	100	011	010	001	000
new state for center cell	0	0	0	1	1	1	1	0

Rule table for Rule 110 CA (110 in binary is 1101110) is also given below:

Table (1.2): Rule Table of Rule 110 CA

current pattern	111	110	101	100	011	010	001	000
new state for center cell	0	1	1	0	1	1	1	0

Rule 30 and rule 110 are interesting, rule 30 produces complex, and seemingly-random patterns from simple well-defined rules, while rule 110 is neither completely stable nor completely chaotic, and a conjecture by Stephen Wolfram was verified by Matthew Cook around 2000 that Rule 110 is Turing complete (i.e. capable of universal computation). Figures (1.11) and (1.12) show space-time diagram for rule 30 and 110

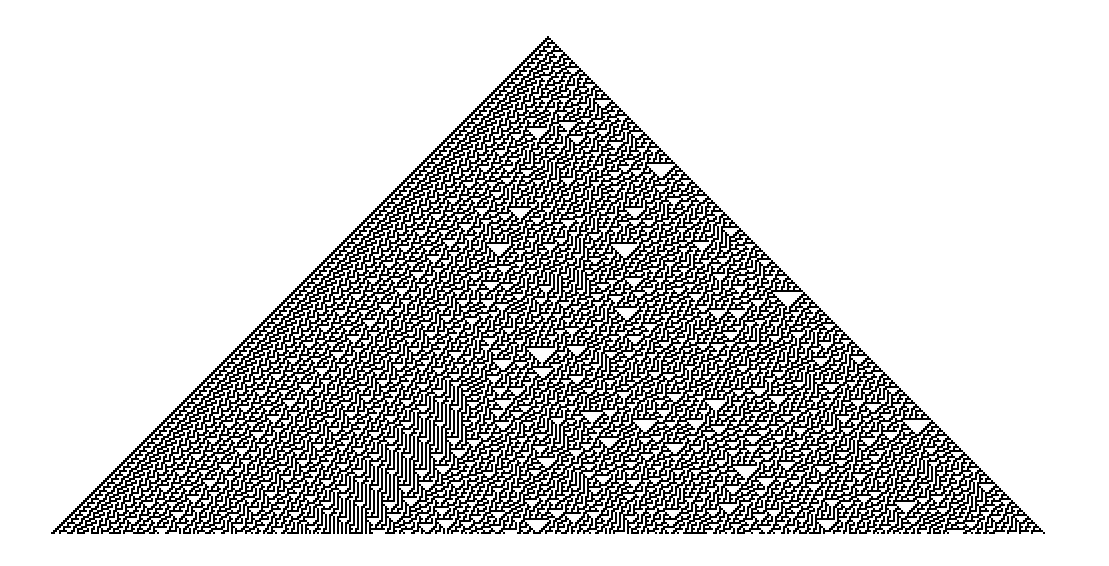


Figure (1.11): Rule 30 space-time diagram

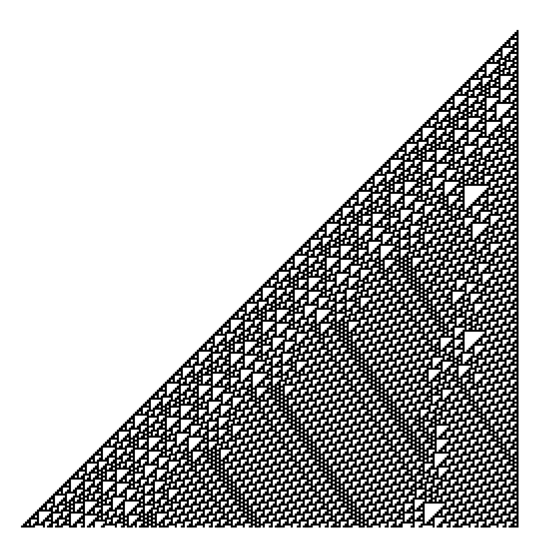


Figure (1.12): Rule 110 space-time diagram

2. Reversible Cellular Automata

CA is reversible if there is exactly one past configuration for every current configuration, reversible CA are used to simulate physical phenomena as gas and fluid dynamics, to decide whether CA is reversible or not, some well known algorithms can be used (Sutner, 1999).

3. Totalistic Cellular Automata

A special class of CA in where the next value of the center cell at time $t+1$ depends only on the sum of its neighbors at time t, with center cell included or not included in the neighborhood. Cell state alphabet L is usually a finite set of integer values. If center cell is not included in the neighborhood, then CA is called *outer totalistic* CA, Conway's Game of Life is an example of outer totalistic CA

1.6.3 Cellular Automata Classes

Stephen Wolfram divided CA into four classes according to their behavior (Wolfram, 1984)

1. Class 1: Limit Points

Almost all initial configurations evolve into a stable, homogeneous state after a finite number of time-steps, all randomness in initial configuration disappears.

2. Class 2: Limit Cycle

This class of CA serves as *filters*; this makes them useful for image processing. Like class1 CA, almost all initial configurations evolve into stable state, but some of the randomness remains not filtered out, the states generated are either stable or periodic with small periods. Examples of this class are rules with numbers 8, 24, 40 and 56 (Wolfram, 1984).

3. Class 3: Chaotic Attractor

This class generates periodic, chaotic patterns from almost all initial configurations, surrounding noise quickly destroys any generated stable structures, the statistical properties of the resulting patterns are similar to statistical results of almost all initial configurations, this class is the most frequent class, rule 18 is an example of chaotic CA.

4. Class 4: More Complex

In most cases CA of this type “dies” (reaches value 0), but in some cases it generates stable or periodic structures after a very large number of iterations, some generated structures are propagating. Game of Life and rule 110 are examples of this class. Wolfram has a conjunction that, almost all, if not all, CA of this class are capable for universal computation, this was proved for Game of Life and rule 110 CA.

1.6.4 Examples of Cellular Automata

The most widely known example of CA, it was introduced by John Conway (Conway, 1976) while he was studying microscopic behavior a population, 100 by 100 lattice size was used (Thomas, 2000), the rules were introduced to describe population growth, many researches refers to it as a mathematical game of CA.

GoL is a two dimensional CA with two states dead “0” and alive “1”, The following rules are defined under Moore neighborhood: If the cell is alive then it stays alive if it has two or three live neighbors (Survival rule), otherwise it dies of loneliness or overcrowding (Death rule). If the cell is dead then it becomes alive if it has exactly three living neighbors (Birth rule).

A dynamic system results by repeating the rules over and over many times to get new generations which exhibits a surprisingly complex behavior. Different patterns occur in GoL, these patterns can be classified in three groups, Stable (or still lives) which remains stable through time, repeating (or oscillators) which repeat themselves every fixed number of generations, and spaceships which move themselves across the lattice. Figures (1.13, 1.14, and 1.15) show some examples of each type.

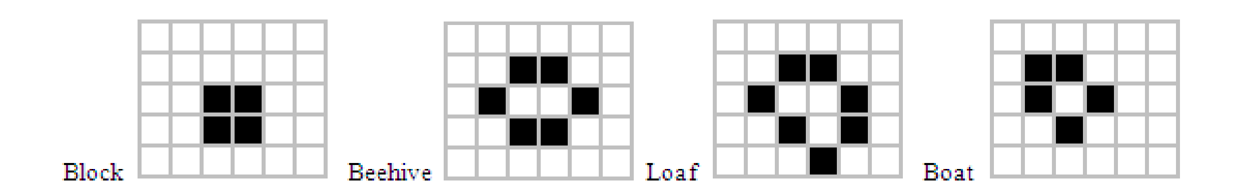


Figure (1.13): Examples on stable (still lives) GoL patterns

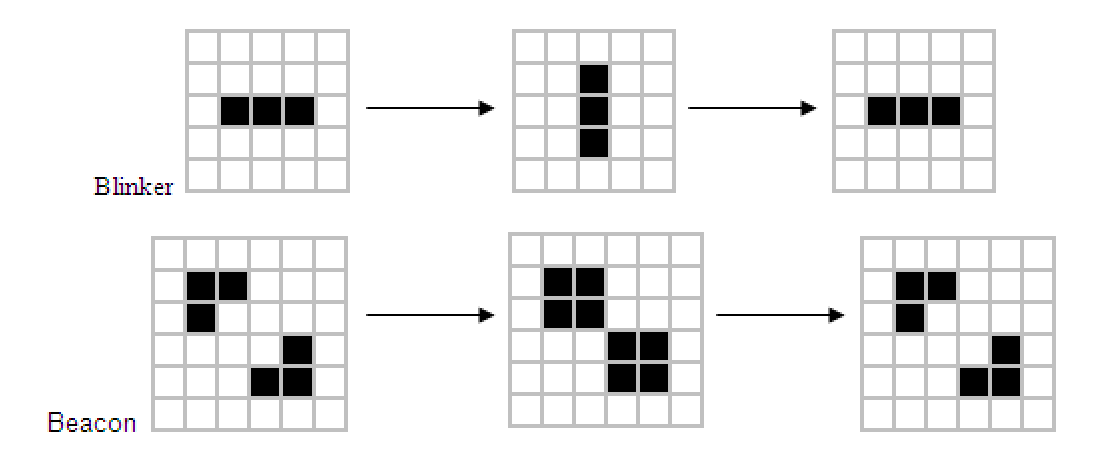


Figure (1.14): Examples on repeating (oscillators) GoL patterns

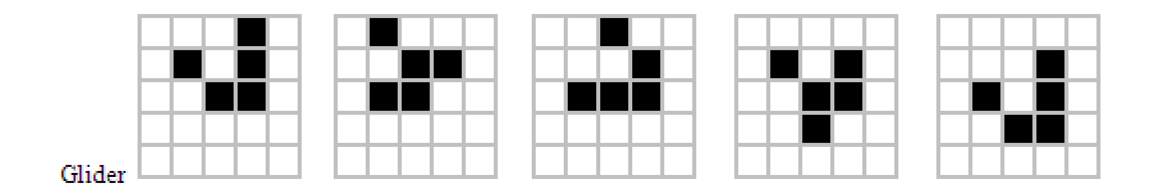


Figure (1.15): Example on moving (spaceships) GoL patterns

1.7 Sequential Search

Sequential searching algorithms are sequential algorithms used to feature selection where you select a subset of existing features.

Examples of sequential searchers:

1. Sequential Forward Selection
2. Sequential Backward Selection
3. Plus-l Minus-r Selection
4. Bidirectional Search
5. Sequential Floating Selection

a. Sequential Floating Forward Search

b. Sequential Floating Backward Search

An objective function is used in sequential searchers, this objective function evaluates the selected subsets and returns a measure of how goodness they are. There are two groups of objective functions which are filters and wrappers.

In filters, information content of subsets is evaluated by objective functions. In wrappers, the objective function evaluates subsets by their predictive accuracy by statistical resampling or cross validation; so objective function here is a pattern classifier. Wrappers are slow, but they are accurate

1.8 Measurement Variables

In this section we will discuss the measurement variables used by literature to evaluate the performance and quality of image processing algorithm, the measurement variables we will discuss here are Hamming Distance (HD), Mean Squared Error (MSE), Peak Signal to Noise Ratio (PSNR), and Exact Similarity (ES).

1. Hamming Distance (HD)

Was originally used to detect and correct errors in digital communication, HD defines the number of different bits between two vectors.

For two m x n images I and K, HD is evaluated by the formula

$$HD = \frac{1}{m\,n}\sum_{i=0}^{m-1}\sum_{j=0}^{n-1} I(i,j) \oplus K(i,j) \quad \dots (1.5)$$

Where the value of $I(i,j) \oplus K(i,j)$ is 1 if it is greater or equal to a given threshold *t*, and 0 otherwise. The minimum HD means the most quality performance results. For example, if threshold value is 10, *I(i,j)* = 214, and *K(i,j)* = 248, then 214 $\oplus$ 248 = 46 (in binary 11010110 $\oplus$ 11111000 = 00101110); and because 46 > 10, then $I(i,j) \oplus K(i,j)$ = 1

2. Mean Squared Error (MSE)

Mean Square Error calculates the average of the "errors" squares, the error is the difference between the value of estimated (restored) result and the original value.

For two *m* x *n* images *I* and *K*, MSE is evaluated by the formula

$$MSE = \frac{1}{m\,n}\sum_{i=0}^{m-1}\sum_{j=0}^{n-1}[I(i,j) - K(i,j)]^2 \quad \ldots (1.6)$$

MSE gives more accurate measurement for performance than HD; it gives the degree of difference between two images, while HD gives the number of different pixels between two images. The minimum MSE means the most quality performance results.

3. Peak Signal to Noise Ratio (PSNR)

An engineering term, describes the ratio of the maximum possible power of a signal and the power of the corrupting noise. PSNR is usually expressed in the form of logarithmic decibel scale.

PSNR is defined via MSE as

$$PSNR = 20 \cdot \log_{10}\left(\frac{MAX}{\sqrt{MSE}}\right) \quad \dots (1.7)$$

Where MAX is the maximum value of the pixel in the image (e.g. MAX=255 for 255-grayscale image), and MSE is evaluated by the formula described earlier. The maximum PSNR means the most quality performance results.

4. Exact Similarity (ES)

ES is a very simple and straight forward performance measurement; it is the percentage between correctly colored pixels to total number of pixels. Sometimes a threshold may be used; so, if the absolute difference between the two compared pixels is within a given threshold, then the pixel is considered correctly colored.

For *m* x *n* image, ES is given by the following formula

$$ES = \frac{\text{Number of correctly colored pixels}}{m\,n} \quad \dots (1.8)$$

The maximum ES value means the most quality performance results. This measurement was used by Guieb, 2007 as an objective function for SFFS (we discussed SFFS in a previous section).

We will use HD, PSNR, and ES to measure and evaluate our results with other algorithms in chapter 4.

Chapter Two

Related Work

2. Related Work

In this chapter, we will show previous work related to image enhancement and restoration using CA, this chapter describes and discuses existing algorithms and tries to analyze them in respect to advantages and disadvantages to build a starting point for further improvement.

An application of two dimensional cellular automata for noise removal and edge detection problems was presented in (Popovici and Popovici, 2002). They compared their results to classical methods such as Gaussian filter and SUSAN method.

In their CA model for noise removal, they proposed a dynamical rule to solve the problem, given a noisy image which forms the initial configuration; the rule produces a trajectory whose final configuration corresponds to a noise-reduced version of the image. One advantage of dynamic rules is that they are desirable to be applied to any kind of images (monochromatic, gray level, or colored images).

An image is a two-dimensional n × n array, each cell in the image is represented by the triplet (i, j, k), where (i, j) is its position in the grid, and k represents its color.

They based their model on a two-dimensional symmetric non-deterministic CA, of the form:

$$\mathcal{A} = (S, N, \delta) \quad \ldots (2.1)$$

With S = {#, 0, 1, …, k-1}, such that they represented a pixel color by a state in {0, 1, …, k-1}, and # to represent the cell outside the grid, N is Von Neumann neighborhood, δ Is the local transition function, and it is based on a criteria to compare

the central cell with its neighbors. Such that $\delta : S^5 \to S$ is defined as:

$$\delta((s_i)_{i=1}^5) = \begin{cases} j\ (\neq \#), & \text{if } s_3 \neq \# \text{ and } |\{i \mid s_i = j\}| = \max_{l=0}^{k-1} |\{s_i = l\}|, \\ \#, & \text{if } s_3 = \#. \end{cases} \quad \ldots (2.2)$$

The transition function above can be summarized as follows: A cell inside the grid changes its state to be the same as the state of the majority of its neighbors, while any cell outside the grid will remain outside the grid (Popovici and Popovici, 2002).

As they used dynamic rule for noise removal, (Popovici and Popovici, 2002) proposed a dynamic rule for two dimensional cellular automata for border detection. The final configuration of this rule must have only the cells which correspond to the borders of the image active.

Their edge detection model is based on two dimensional cellular automaton

$$\mathcal{A} = (S, N, \delta) \quad \ldots (2.3)$$

With S = {#, 0, 1, …, k-1}, N is Von Neumann neighborhood, δ Is the local transition function, such that $\delta : S^5 \to S$ is defined as:

$$\delta(s_1, s_2, s, s_3, s_4) = \begin{cases} 0, & \text{if } |s - s_i| < \varepsilon,\ (\forall)i = \overline{1,4}, \\ s, & \text{otherwise} \end{cases}, \quad \ldots (2.4)$$

ε Is a threshold value, which is chosen according to the application.

The transition rule compares the central cell with the state of each of its neighbors, if the difference between the state of the central cell and the state of any of its neighbors

is less than a given threshold, then this cell is not considered as an edge, otherwise the center cell is an active edge.

This rule is independent of the nature considered image; so it can be applied to any kind of image (monochromatic, gray level, or colored images).

The result of their proposed cellular automata rule shows edge connectivity at junctions, do not detect false edges, and shows qualitative similarity with the results of SUSAN edge detector.

From the results obtained by (Popovici and Popovici, 2002) proposed cellular automata rules for noise removal and edge detection, we can conclude that introducing a very simple cellular automata transition rule may give very efficient results. Image processing using cellular automata can be independent from the characteristics of the processed image, and the introduction of threshold in a dynamic transition rules may be satisfactory for some applications.

A more complex CA algorithm was introduced by (Guieb, 2007), this algorithm tries to train Cellular Automata to remove noise from images, it uses sequential floating forward search SFFS (discussed in previous section) to select a subset of rules, which is supposed to give best results, and then apply them to the image. Methodology of (Guieb, 2007) consists of 6 parts, as follows:

1. Program Input

Program input is the original noiseless image, and the noisy image. The noisy image is the initial configuration for Cellular Automata. Both images are converted to binary images according to a threshold given by the user.

2. Cellular Automata States, Neighborhood, and Rules

Since Moore's neighborhood is used, and processing is done on binary images; 2^8 = 256 neighborhood configurations exist, but taking into account the 90 degrees rotational symmetry and bilateral reflection, this decreases the number of neighborhood configurations to 51.

When a configuration matches the state of the central cell and its neighbors, the state of the central cell is reversed.

3. Generation of Rules

Sequential Floating Forward Search (SFFS) method was used to generate rules, using the following algorithm.

Algorithm 2.1 (Guieb, 2007) CA

Definitions:

J: Objective function

Y: Combination of selected rules

x: the rule to be added or subtracted

Procedure:

1. Start with the empty set Y
2. Select the best rule

   x+ = argmax[J(Yk + x)]

   Yk = Yk + x+ ; k = k + 1

3. Select the worst rule

x- = argmax[J(Yk - x)]

4. If J(Yk – x-) > J(Yk) then

    Yk+1 = Yk – x ; k = k + 1

    Go to step 3

   Else

    Go to step 2

SFFS will stop only when the fittest objective function is computed.

$$\text{Objective Function} = \frac{\text{Number of correctly colored pixels}}{\text{Total number of pixels}} \quad \ldots (2.5)$$

4. First Extension

The generated set is applied to the noisy image until the objective function is no longer improved.

5. Second Extension

Using the output of the previous run, and after inverting the color of the rules from black to white or from white to black, CA and SFFS process is executed again.

6. Comparison of Results

Although CA introduced by (Guieb, 2007) shows superior results compared to opening and closing techniques, the running time is a major problem for the introduced CA; because SFFS is used to select the best rules which results in the best objective function.

This large execution time indicates that using SFFS is not recommended to choose the best rules for CA, and it will be avoided in our research.

More efficient CA algorithm was introduced in An Effective Filtering Algorithm for Image Salt-pepper Noises Based on Cellular Automata, by (Liu, et al. 2008), they introduced specialized rules which handle and remove salt and pepper noise, they considered salt-pepper noise density as an image feature to decide which neighborhood to use; so, when the density is high, they use Moore neighborhood.

CA rule introduced is as follows:

Algorithm 2.2 (Liu, et al. 2008) CA

Procedure:

1. In Moore neighborhood, calculate the minimum state value s_{min} and maximum state value s_{max} , if the state of the current cell satisfies $s_{min} < s^{t}_{i,j} < s_{max}$, then $s^{t+1}_{i,j} = s^{t}_{i,j}$, otherwise go to step 2
2. If $s_{max} = s_{min}$, or these cells in the moore neighborhood have only two states s_{max} and s_{min}, then go to step 3, otherwise go to step 4
3. If $s_{min} \neq 0$, then $s^{t+1}_{i,j} = s_{min}$, if $s_{max} \neq 255$, then $s^{t+1}_{i,j} = s_{max}$, otherwise $s^{t+1}_{i,j} = s^{t}_{i,j}$
4. Except the cells of s_{min} and s_{max} in the Moore neighborhood, compute the mean value s_{mean} of all other cells, if $abs(s^{t}_{i,j} - s_{mean}) <$ threshold, then $s^{t+1}_{i,j} = s^{t}_{i,j}$, otherwise $s^{t+1}_{i,j} = s_{mean}$

They performed experiments on the classical Lena image of size 256x256, and evaluated their results using the hamming distance between restored and original image.

$$HD = \frac{1}{L}\sum_{i=1}^{L} x_i \oplus y_i$$, Where L is the number of pixels in the image.

If the result of applying (Liu, et al. 2008) on noised image is not good enough, (Liu, et al. 2008) can be applied more than once to get more satisfying restored image.

The proposed CA by (Liu, et al. 2008) is a very good CA, it removes salt and pepper noise very efficiently, the advantage of the proposed method becomes more obvious, as the density of the noise increases, especially for noise density bigger than 0.4

Iteration feature is an advantage for Liu, et al. (2008) CA; if the noise ratio is very high, then most of the cells in Moore neighborhood maybe noise cells, this will affect the final result, but if the iteration feature is used, then the noise will be removed effectively.

For our research, it will be a very good advantage to make our proposed method iterative, and to provide rules which are adaptive to each type of noise.

Selvapeter and Hordijk (2009) Introduced "Majority" CA update rule to remove noise from binary and 256-levels gray scale images in their paper Cellular Automata for Image Noise Filtering, they used Moore neighborhood for binary images, and Von Neumann neighborhood for both binary and gray scale images. The update rule was not applied to boundary image pixels (Fixed value boundary condition).

The Majority CA rule is as follows:

Algorithm 2.3 (Selvapeter and Hordijk 2009) "Majority" CA

Procedure:

1. If the center pixel is black or white (0 or 255) then
   a. If there is a gray level that is a majority in the neighborhood, replace the center pixel with this gray level.
   b. Else, if there is a tie (no gray level has a majority), then go to step 2.
2. Chose either deterministically or randomly to chose the pixel that will replace the central pixel
   a. Deterministically: replace the central pixel with a pixel in a fixed position in the neighborhood (e.g. the pixel on top of the central pixel).
   b. Randomly: replace the central pixel with any randomly chosen pixel from the neighborhood.

Using the peak signal to noise ratio (PSNR), they compared the results of their CA rule with standard filtering techniques like Median 3x3, Median 5x5, Switching I, Switching II, and Adaptive median, applying the rule on one binary image "Camera-man" and two gray level images "Fishing-boat" and "Lena" which are noised with salt and pepper noise.

By comparing the obtained results of “Majority” CA with other techniques, for low noise ratio, the results of Switching II algorithm are close to "Majority" CA, but as noise ratio increases, the result of "Majority" CA becomes better. And by comparing “Majority” CA with Median filter, the blurring effect of Median filter was obvious as the window size increases; this caused the value of PSNR to be degraded.

Results of the new Mirrored “Majority” CA were improved by considering the boundary cells in their CA (an existing neighbor for the boundary cell was mirrored into the corresponding not existing neighbor cell).

One more notice which given an advantage for “Majority” CA over Switching I, Switching II, and PSM impulse detectors, is that for Switching I, Switching II, and PSM impulse detectors the noise was not able to be detected in some areas where gray levels are comparable to noise levels.

"Majority" CA match Popovici and Popovici (2002) CA in that it introduce simple CA rules that can outperform the results of traditional filtering techniques; also, some specific rules can be made to handle specific type of issues like blurring and boundary effect.

After analyzing and testing "Majority" CA, we can conclude that Random "Majority" CA with Von-Neumann neighborhood is the best technique among "Majority" CA techniques, and that it is possible to improve "Majority" CA by, firstly, providing new criteria of selecting the gray level that will replace the central cell to adapt with the noise ratio (i.e. not always choosing the gray level that is a majority in the neighborhood), and secondly, enhance the Deterministic and Random techniques which are used if there is a tie, and finally, by applying iterative solution rather than one iteration.

An Enhanced Cellular Automata for Image Noise Removal was introduced by (Abu dalhoum, et al. 2011), their CA deals with both salt and pepper noise and uniform noise, first they detect the type of the noise by computing the histogram of the noise image, then they apply the CA transition rules shown in algorithm 2.4

Algorithm 2.4 (Abu dalhoum, et al. 2011) CA

Procedure:

Check value of current cell Xi,j and values of its neighbor.

1. If uniform noise
   a. mx=max(neighbors);
   b. mn=min(neighbors);
   c. if Xi,j ==mx or Xi,j ==mn, take the median value of the neighbors y=SMij;
2. If salt and pepper noise
   a. if (Xi,j ==0) or (Xi,j ==255) and there are neighbors that are not 0 nor 255, Xi,j = The median of step 3.a;
   b. else, Xi,j= mean(Neighbors);

If the noise is uniform, the median value is calculated after excluding cells with maximum and minimum states from the neighborhood, and then the current cell is assigned the calculated median value. If the noise is salt and pepper noise, and the center cell is black or white, the median value is calculated after excluding black and white cells from the neighborhood and assigned to the current cell, but if all cells in neighborhood are black or white, the mean value is calculated for them and assigned to the current cell.

To compare their results, (Abu dalhoum, et al. 2011) tested their CA algorithm on two standard test images, namely, Lena and Boats, and on one other image, Jan image. They compared their results on uniform noise with (Chang, et al. 2008) and compared their results on salt and pepper noise with (Liu, et al. 2008) and used MSE and HD as performance measurements. The results obtained shows that the CA model for uniform noise produces results are better than (Chang, et al. 2008), but for salt and pepper noise,

the results are close to (Liu, et al. 2008). In this CA model also, it is obvious that introducing CA transition rules which are specific for one type of noise gives better results.

Chapter Three

Research Environment

And

Proposed Techniques

3. Research Environment and Proposed Techniques

In this chapter we describe research environment and techniques we used in this thesis, and then we will present our proposed CA model.

3.1 Experiment Environment

We used the well known MATLAB software 7.9 to implement CA models, and we executed the implemented programs on HP Compaq 6710b laptop running Windows XP Professional SP3 with the following specifications

Table (3.1): Hardware specification

Component	Specification
CPU	Intel Core2 Duo CPU T7700, 2.40 GHz
Motherboard	Mobile Intel® GM965
Memory	DDR2 SDRAM, 667-MHz, two slots supporting dual channel memory, 2048-MB SODIMM
Hard drive	SATA 160-GB 5400 rpm, HP 3D DriveGuard
Graphics	Intel GMA X3100, up to 384-MB shared system memory

3.2 Proposed Techniques

We have designed a CA model that is composed of two parts for image noise removal, our CA model handles both uniform noise, and salt and pepper noise, the algorithm gets the type of noise as an input, the proposed model is two dimensional CA model. Here we discuss the proposed model and show its characteristics and advantages.

3.2.1 Iterative Weighted Edges CA

This proposed CA model consists of two parts, the first one, we call, Iterative CA, and the second part is Weighted Edges CA, both parts use Moore neighborhood and adopt an improved technique to select or calculate the value which will replace the state of the cell in the center, one more advantage is that they try to decrease, as much as possible, the effect of noised cells among the neighborhood on the final result.

The most important characteristic for the proposed Iterative CA and from which its name comes is its iterative behavior, where it avoids to update the value of noised center cell for the time t+1 if the neighborhood at time t contains data that is not enough to calculate an accurate result; so, it leaves the value as it is until the next iteration comes, and on the next iteration (at time t+1), the CA hopes that the changes occurred to the neighboring cells in the last iteration (at time t) will cause to obtain enough information to evaluate an accurate result.

Figure (3.1) summarizes the idea of leaving the center cell state untouched until the next iteration comes.

0	126	121
130	255	0
255	0	255

Figure (3.1) Neighborhood at time t contains high noise level

Noise percentage in the Moore neighborhood around the center cell is high (almost 66%); so the state of center cell will be remain 255 for t+1. Then, at time t+1, if the

neighborhood was less noisy, as shown in Figure (3.2), CA will calculate a value that is most probably accurate and use it for central cell.

124	126	121
130	255	122
129	0	255

Figure (3.2) Neighborhood at time t+1 contains low noise level

The second part, Weighted Edges CA, uses the same algorithm of Iterative CA, but it gives more weight to the cells that lies on an edge. The output of Iterative CA is processed twice using Canny edge detection method to get what we call "double edge image" this double edge image is used by Weighted Edges CA to determine which cells are on edges; so, they will be given more weight. The diagram shown in Figure (3.3) shows the flow of Iterative Weighted Edges CA.

Because of parallelism of CA, time complexity of Iterative CA and Weighted Edges CA is O(1).

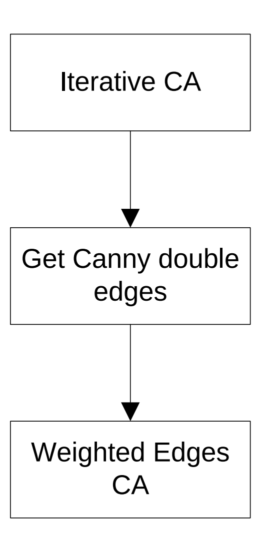


Figure (3.3) Flow diagram of Iterative Weighted Edges CA

The following two listings show the pseudo code of Iterative CA and Weighted Edges CA respectively.

Algorithm 3.1 Iterative CA

Input: Noised image, noise type

Output: Restored result image

Preprocessing: Regenerate the input noise image by mirroring its borders

Procedure:

For each cell in the input image, except the cells in mirrored borders, do the following:

A. If Salt and Pepper noise

1. If the state of current cell s^t is not 0 or 255, $s^{t+1} = s^t$
2. For Moore neighborhood, with current cell included in the neighborhood, remove all cells with states 0 or 255, call the result ‘filtered neighborhood’.
3. If filtered neighborhood is empty, $s^{t+1} = s^t$
4. Get max and min values in filtered neighborhood
5. If the absolute difference between max and min <= 10, s^{t+1} = mean(filtered neighborhood), else, s^{t+1} = median(filtered neighborhood)

B. Else [Uniform noise]

1. Get max and min values in Moore neighborhood (with current cell included in the neighborhood)
2. If state of current cell s^t not equal min nor max, $s^{t+1} = s^t$
3. For Moore neighborhood, remove all cells with states equal to min or max, call the result ‘filtered neighborhood’.
4. If filtered neighborhood is empty, $s^{t+1} = s^t$
5. If the absolute difference between max and min <= 10, s^{t+1} = mean(filtered neighborhood), else, s^{t+1} = median(filtered neighborhood)

The algorithm tends to calculate the mean value to be the value for the center cell at time t+1 if the values in the neighborhood are close to each other, this was observed experimentally that the mean value is more accurate than median for close values in neighborhood. Otherwise, the median value is calculated and set to the center cell.

Algorithm 3.2 Weighted Edges CA

Input: Noised image, noise type, and canny double edge image of Iterative CA result

Output: Denoised result image

Preprocessing: Regenerate the input noise image, and double edge image by mirroring their borders

Procedure:

For each cell in the input image, except the cells in mirrored borders, do the following:

A. If Salt and Pepper noise

1. If the state of current cell s^t is not 0 or 255, $s^{t+1} = s^t$
2. If the current cell is on an edge, and the number of cells in Moore neighborhood that are on an edge > 2, replicate the cells that are on edges (now the neighborhood contains more than 9 cells, call it ‘weighted neighborhood’).
3. For the weighted neighborhood, remove all cells with states 0 or 255, call the result ‘filtered neighborhood’.
4. If filtered neighborhood is empty, $s^{t+1} = s^t$
5. Get max and min values in filtered neighborhood
6. If the absolute difference between max and min <= 10, s^{t+1} = mean(filtered neighborhood), else, s^{t+1} = median(filtered neighborhood)

B. Else [Uniform noise]

1. Get max and min values in Moore neighborhood (with current cell included in the neighborhood)
2. If state of current cell s^t not equal min nor max, $s^{t+1} = s^t$
3. If the current cell is on an edge, and the number of cells in Moore neighborhood that are on an edge > 2, replicate the cells that are on edges (now the neighborhood contains more than 9 cells, call it 'weighted neighborhood').
4. For the weighted neighborhood, remove all cells with states equal to min or max, call the result 'filtered neighborhood'.
5. If filtered neighborhood is empty, $s^{t+1} = s^t$
6. If the absolute difference between max and min <= 10, s^{t+1} = mean(filtered neighborhood), else, s^{t+1} = median(filtered neighborhood)

By studying Iterative CA, and comparing it with CAs in literature presented before, Liu et, al (2008) and Selvapeter and Hordijk (2009), we observed that our Iterative CA produced more satisfactory results that the other two CAs, but we also observed that Iterative CA and Liu et, al (2008) give not satisfactory results for cells which lie on edges or near edges. Figure (3.4) shows the cells with maximum HD in result denoised images, and it is obvious that most of these cells lie on edges.

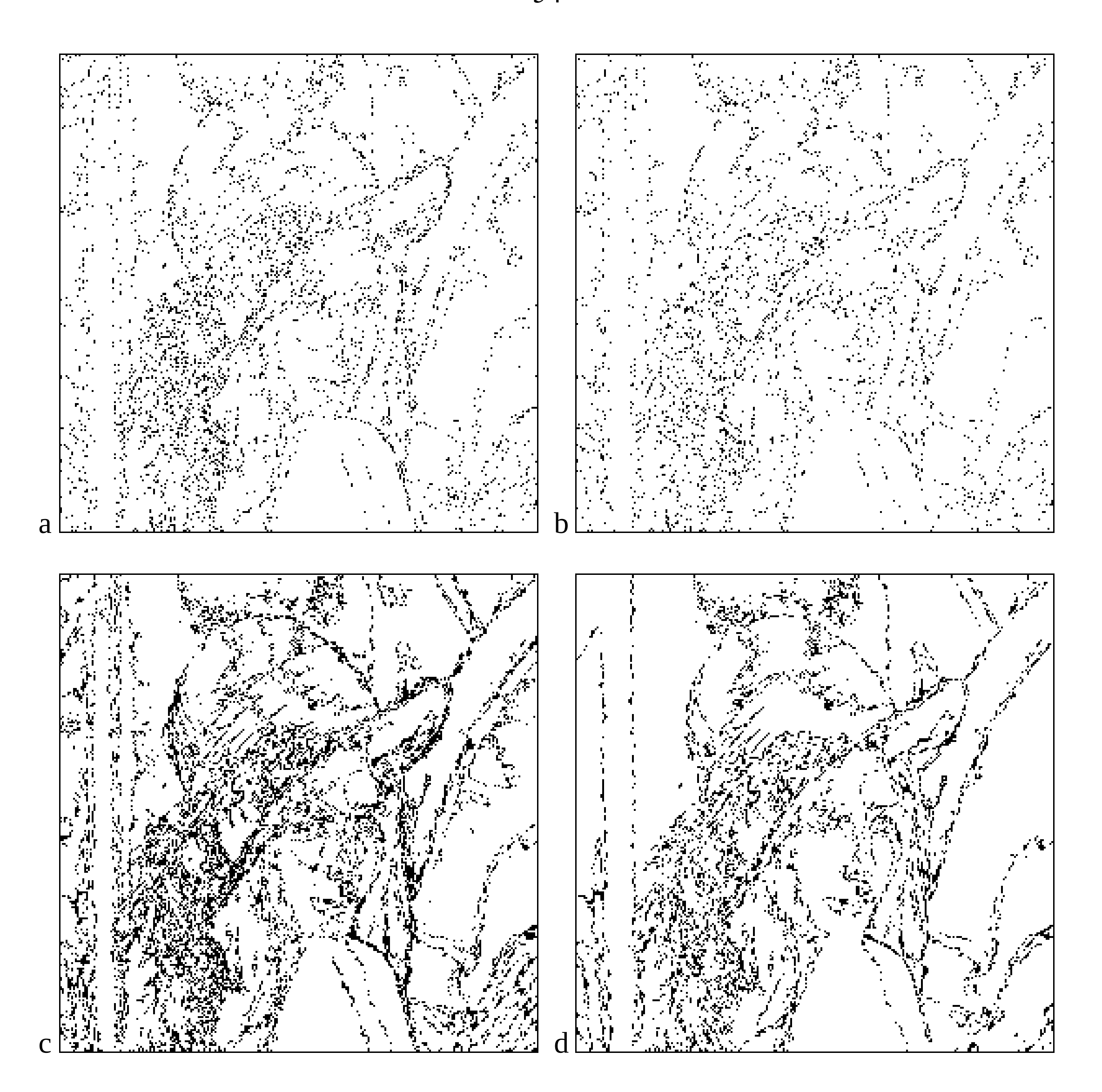


Figure (3.4) Maximum HD values

a: Cells with maximum HD of Liu, et al. (2008) result on 40% noised Lena image

b: Cells with maximum HD of Iterative CA result on 40% noised Lena image

c: Cells with maximum HD of Liu, et al. (2008) 5 iterations result on 80% noised Lena image

d: Cells with maximum HD of Iterative CA 5 iterations result on 80% noised Lena image

Now, if we apply canny edge detection method twice on the result of Iterative CA we get the image shown in Figure (3.5), then we apply Weighted Edges CA, we get a result that is more satisfactory on edge cells. Figure (3.6) shows the final result and shows cells with maximum HD for it.



Figure (3.5) canny double edges result

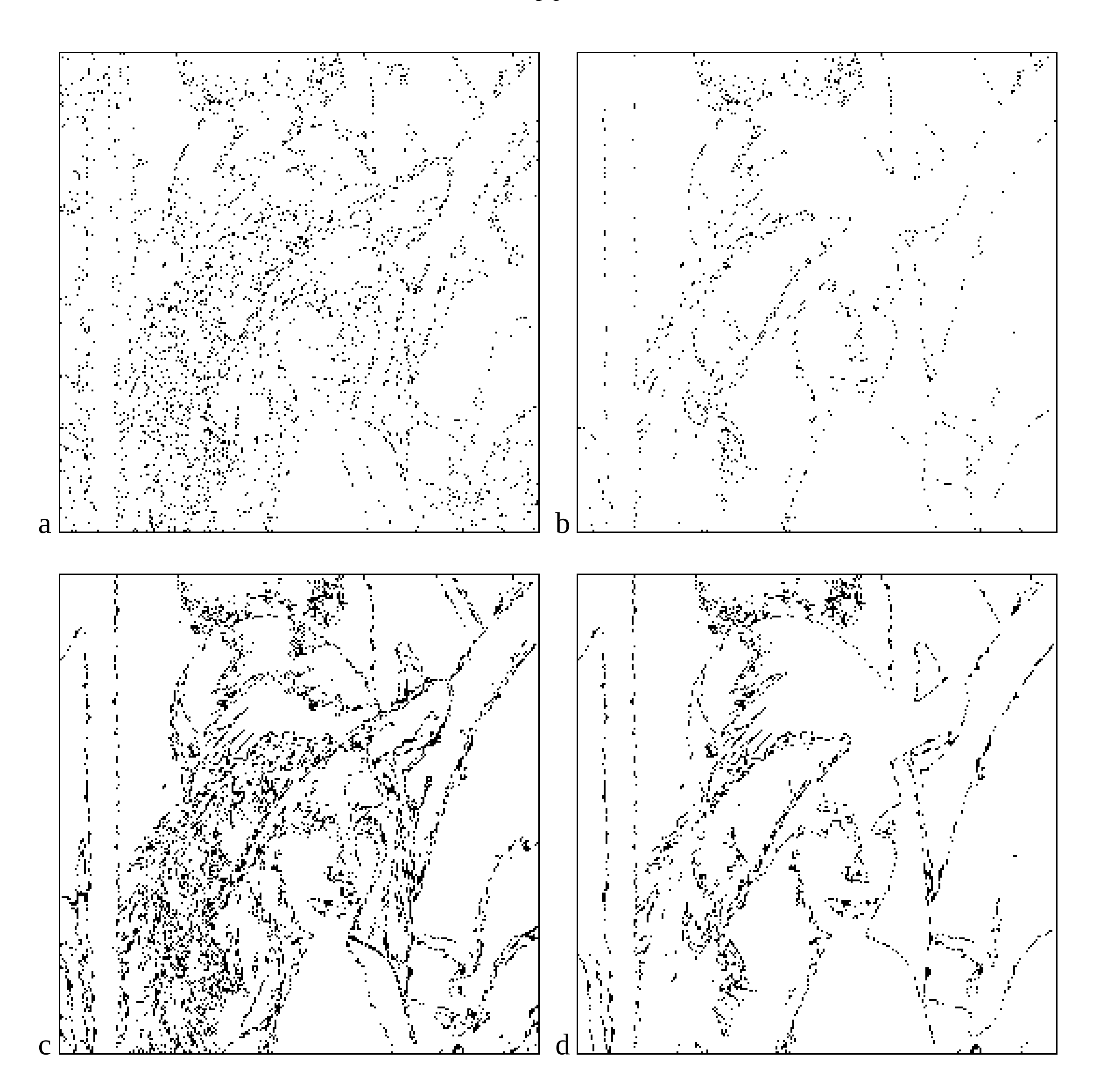


Figure (3.6) Maximum HD values

a: Cells with maximum HD of Iterative CA result on 40% noised Lena image

b: Cells with maximum HD of Iterative Weighted Edges result on 40% noised Lena image

c: Cells with maximum HD of Iterative CA 5 iterations result on 80% noised Lena image

d: Cells with maximum HD of Iterative Weighted Edges CA 5 iterations result on 80% noised Lena image

It is very obvious now that Iterative Weighted Edges CA gives more satisfying results for cells which lie on edges, and this contributes on the overall result of restoring the noised images, and gives more satisfactory results.

Chapter Four

Experimental Results

and

Conclusions

4. Experimental Results and Conclusions

For our expirements we developed a Matlab application for the proposed Iterative Weighted Edges CA in addition to the previously developed MATLAB applications for (Liu et, al. 2008) CA, (Selvapeter and Hordijk, 2009) “Majority” CA, and (Abu dalhoum, et al. 2011) CA, we also used standard median filter in MATLAB to compare our results with median filter.

We executed each method for the number of iterations which gives the best HD, except that for median filter we executed it only once for all noise ratios (one iteration); so it is possible to find that for the same noise ratio, (Liu et, al. 2008) CA, for example, needed 5 iterations to give the best result, while our Iterative Weighted Edges CA needed 4 iterations to do so.

We used three standard measurement variables to compare the results of our proposed CA with other models, these measurement variables are HD, PSNR, and ES. Figure (4.1) shows 4 standard test images that we performed our experiments on, the images were selected from different fields like medicine, and Meteorology fields, in addition to images from nature, these images are: Lena, blown_ic_hr, Katrina_2005, and Skull,.



Figure (4.1) Test images
a: Lena **b:** Wooden-boat **c:** blown_ic_hr **d:** katrina_2005

4.1 Experimental Results for Salt and Pepper Noise

The first set of experiments we executed on salt and pepper noise, the following Figures show the results we got in our experiments and show comparisons between our proposed IWE CA and other CA models.

Figure (4.2) shows results of applying IWE CA to restore Lena image noised with 20%, 40%, and 80% Salt & Pepper noise.



Figure (4.2) Results of IWE CA on Lena test image
a: 20% Salt & Pepper noised Lena image **b:** restored image of a using IWE CA
c: 40% Salt & Pepper noised Lena image **d:** restored image of c using IWE CA
e: 80% Salt & Pepper noised Lena image **f:** restored image of e using IWE CA

In the following Figures, the labels HD/PSNR/ES Median, HD/PSNR/ES Majority CA, HD/PSNR/ES Liu CA, HD/PSNR/ES AD CA, and HD/PSNR/ES IWE CA refer to calculated HD, PSNR, or ES value for the results of Median filter, (Selvapeter and Hordijk, 2009) "Majority" CA, (Liu, et al. 2008) CA, (Abu dalhoum, et al. 2011) CA, and our proposed IWE CA respectively.

Figure (4.3) shows comparison between HD result values for different algorithms on applying them on Lena test image with different noise ratios.

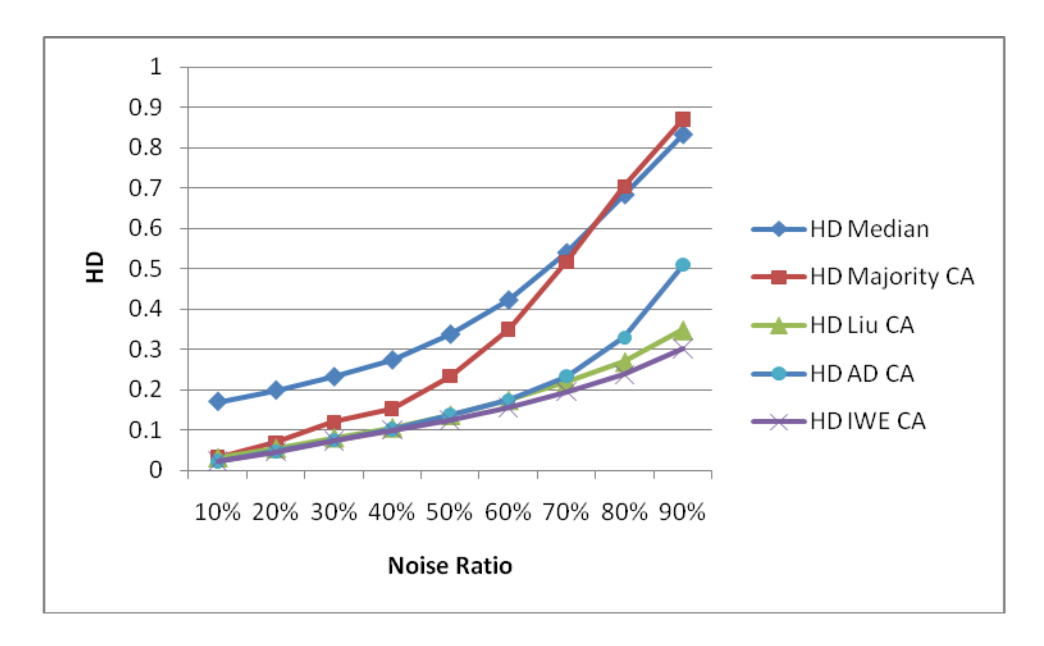


Figure (4.3) HD comparison results of applying different models on Lena test image

As described in section 1.8, the minimum value of HD means better performance for the algorithm in restoring noisy image. It is clear here that our proposed IWE CA results in the minimum HD value, which means that it has the best performance. HD values of (Liu, et al. 2008) and (Abu dalhoum, et al. 2011) are close to each other and to IWE CA for low noise ratios, but for high noise ratios, IWE CA gives best results then comes (Liu, et al. 2008), then (Abu dalhoum, et al. 2011).

Figure (4.4) shows comparison between PSNR result values for different algorithms on different noise ratios added to Lena test image.

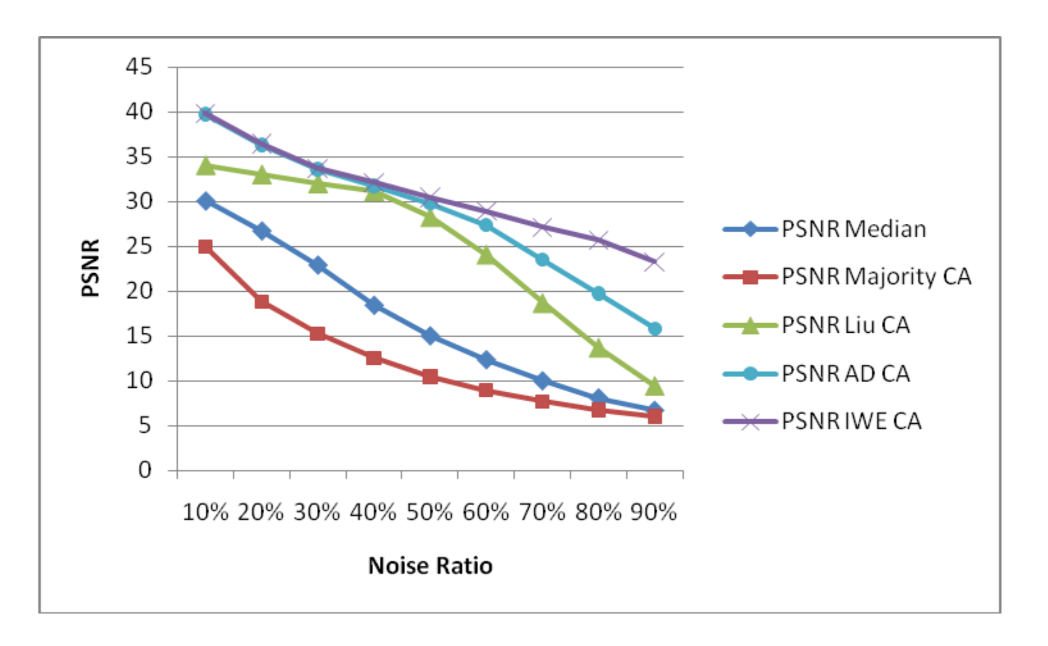


Figure (4.4) PSNR comparison results of applying different models on Lena test image

Our proposed IWE CA gives the maximum PSNR values over different noise ratios, and as the maximum value of PSNR means better performance; this means that IWE CA gives the best performance, then comes (Abu dalhoum, et al. 2011), then (Liu, et al. 2008).

Figure (4.5) shows ES comparison between different algorithms on different noise ratios added to Lena test image.

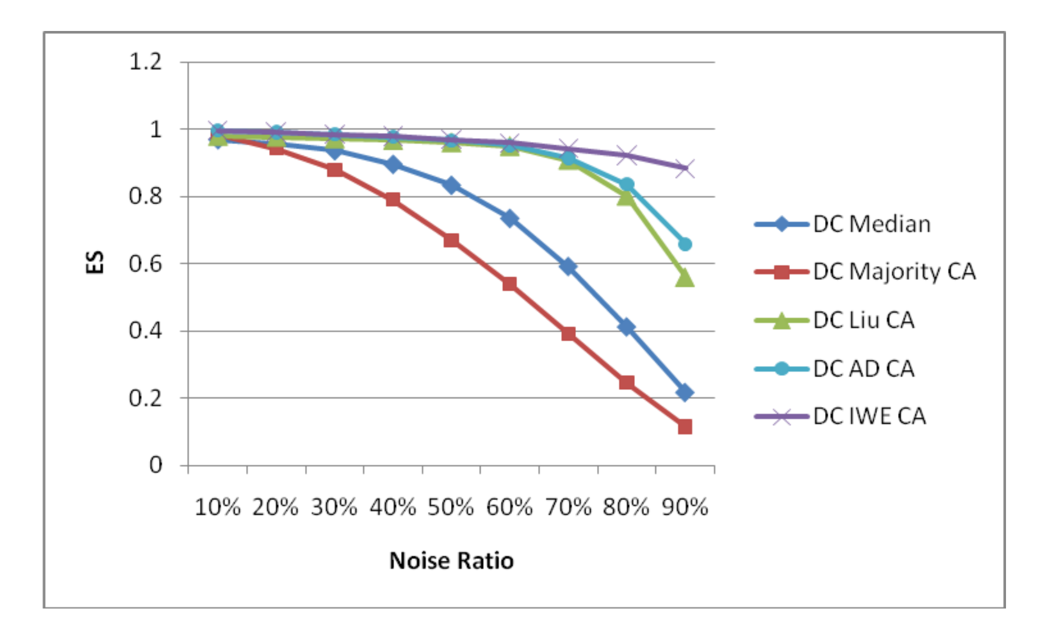


Figure (4.5) ES comparison results of applying different models on Lena test image

For ES, the maximum value means better performance. As Figure (4.5) shows, for low noise ratios ES values for all algorithms are close to each other, but as noise increases, Median filter and (Selvapeter and Hordijk, 2009) “Majority” CA results degrades dramatically, but ES values for (Liu, et al. 2008) CA, (Abu dalhoum, et al. 2011) CA, and our proposed IWE CA remains close, and for noise ratios greater than 70% it becomes obvious that our proposed IWE CA gives the best results.

Figure (4.6) shows results of applying IWE CA to restore Wooden-boat image noised with 20%, 40%, and 80% Salt & Pepper noise.

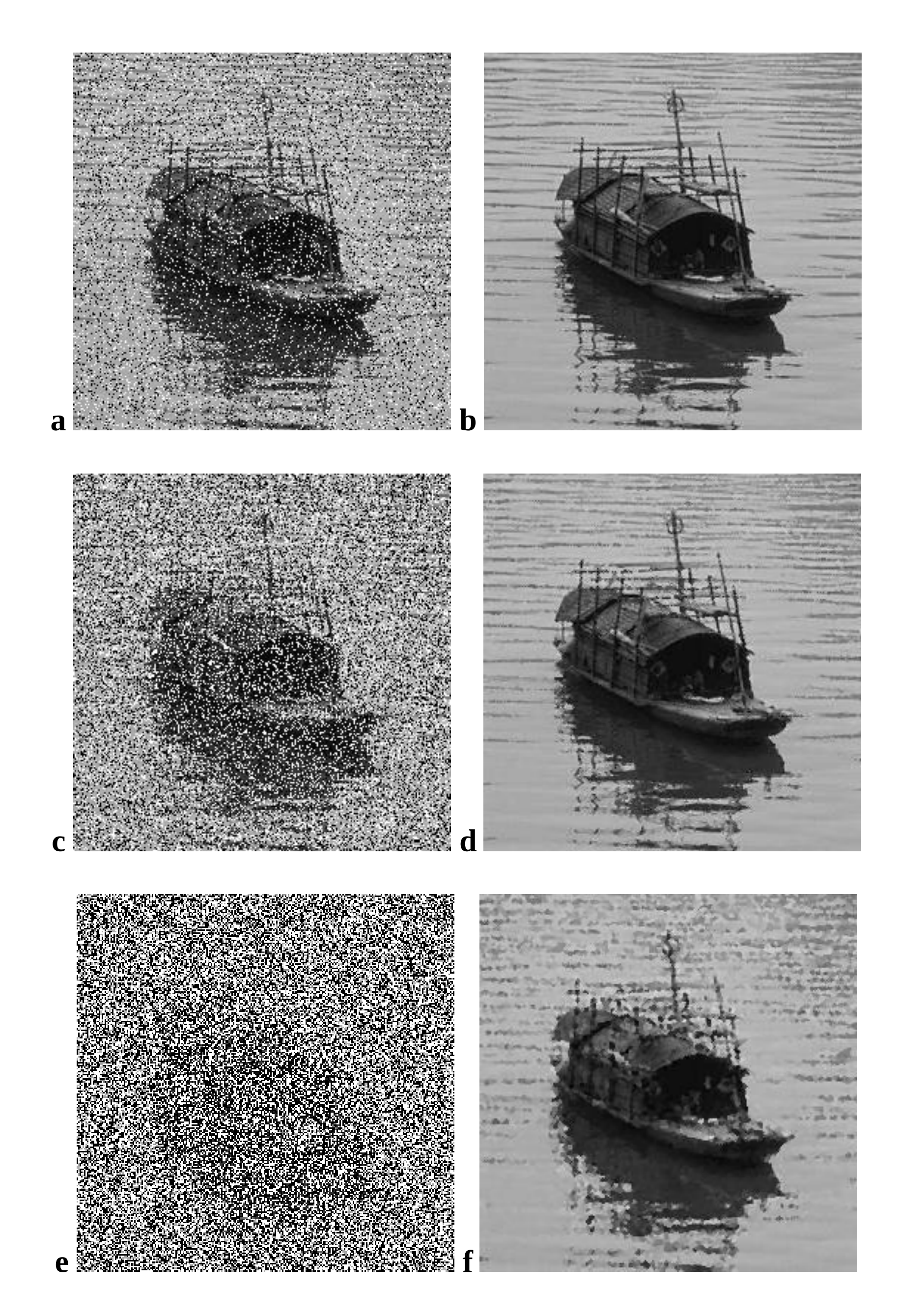


Figure (4.6) Results of IWE CA on Wooden-boat test image
a: 20% Salt & Pepper noised Wooden-boat image **b:** restored image of a using IWE CA
c: 40% Salt & Pepper noised Wooden-boat image **d:** restored image of c using IWE CA
e: 80% Salt & Pepper noised Wooden-boat image **f:** restored image of e using IWE CA

Figure (4.7) shows HD comparison for Wooden-boat test image with different noise ratios.

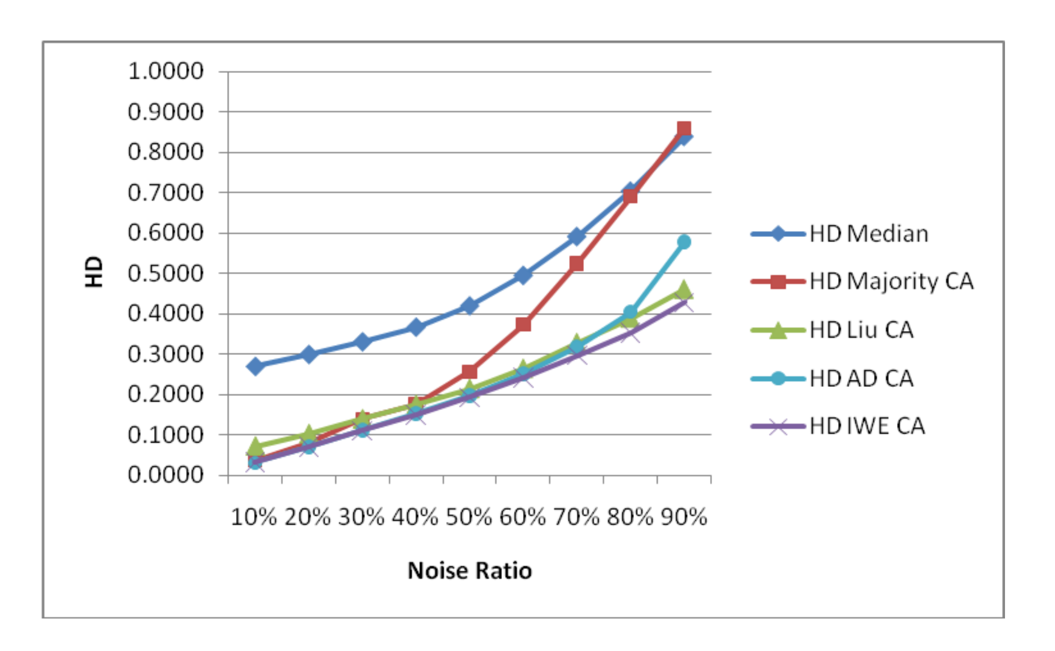


Figure (4.7) HD comparison results of applying different models on Wooden-boat test image

Our proposed IWE CA gives the lowest HD value for different noise ratio, which means best performance.

Figure (4.8) shows PSNR comparison for Wooden-boat test image with different noise ratios.

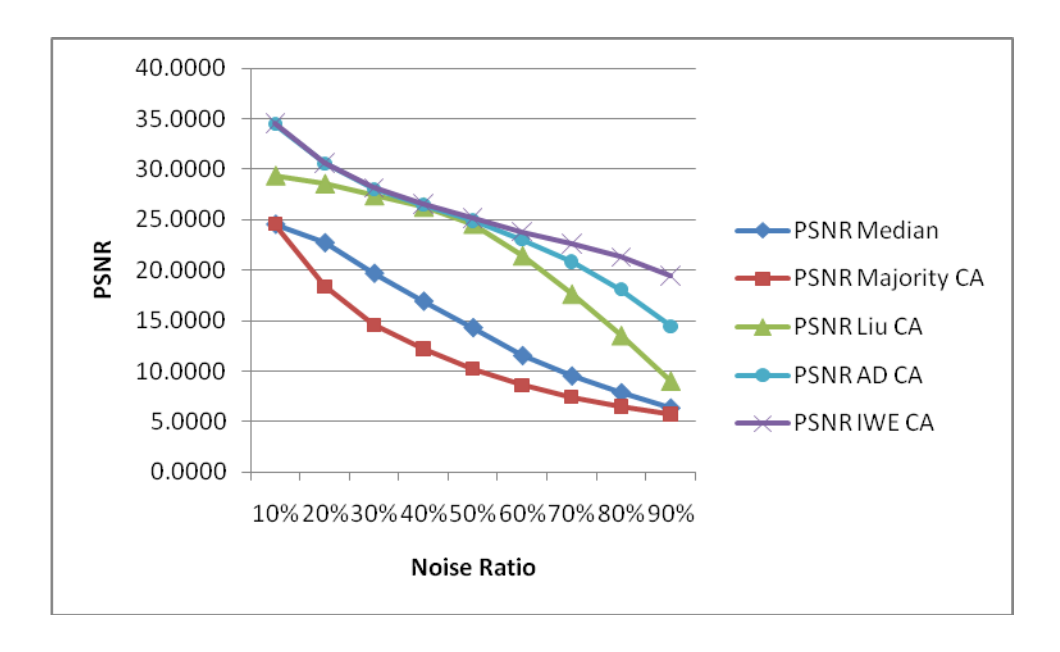


Figure (4.8) PSNR comparison results of applying different models on Wooden-boat test image

IWE CA gives the maximum PSNR value for different noise ratio, which means best performance, then comes (Abu dalhoum, et al. 2011) CA, then (Liu, et al. 2008) CA.

Figure (4.9) shows ES comparison for Wooden-boat test image with different noise ratios.

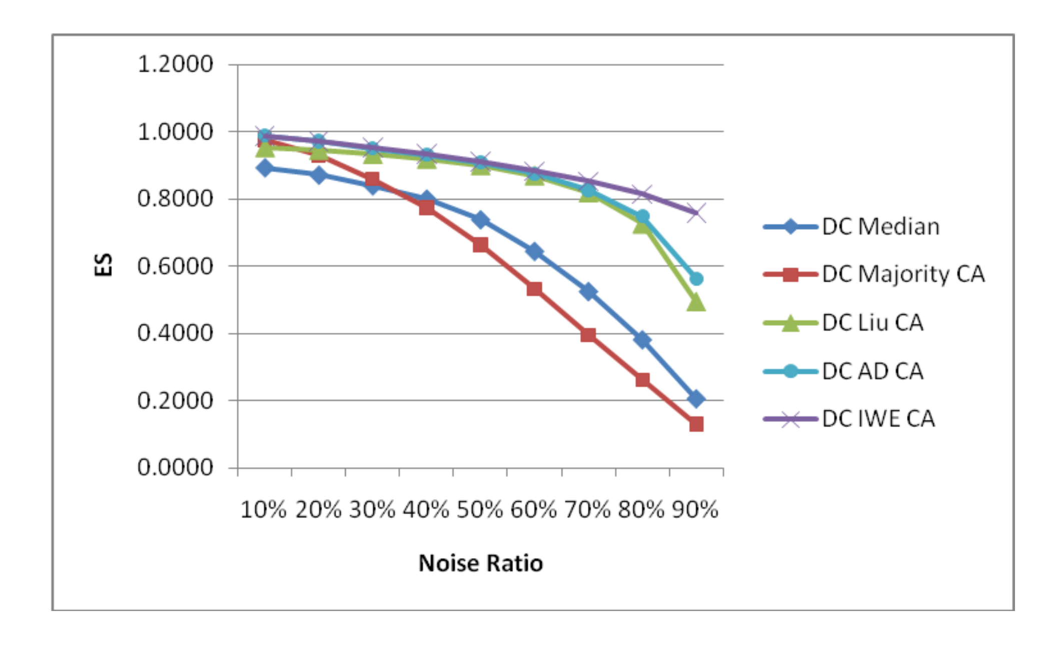


Figure (4.9) ES comparison results of applying different models on Wooden-boat test image

IWE CA, (Abu dalhoum, et al. 2011) CA, and (Liu, et al. 2008) CA give approximately similar results for low noise ratio, but our IWE CA performs better for high noise ratios.

Similarly, Figures (4.10) through (4.15) compare HD, PSNR, and ES for blown_ic_hr and katrina_2005 test images, the same discussed results for Lena and Wooden-boat test images apply on both blown_ic_hr and katrina_2005.

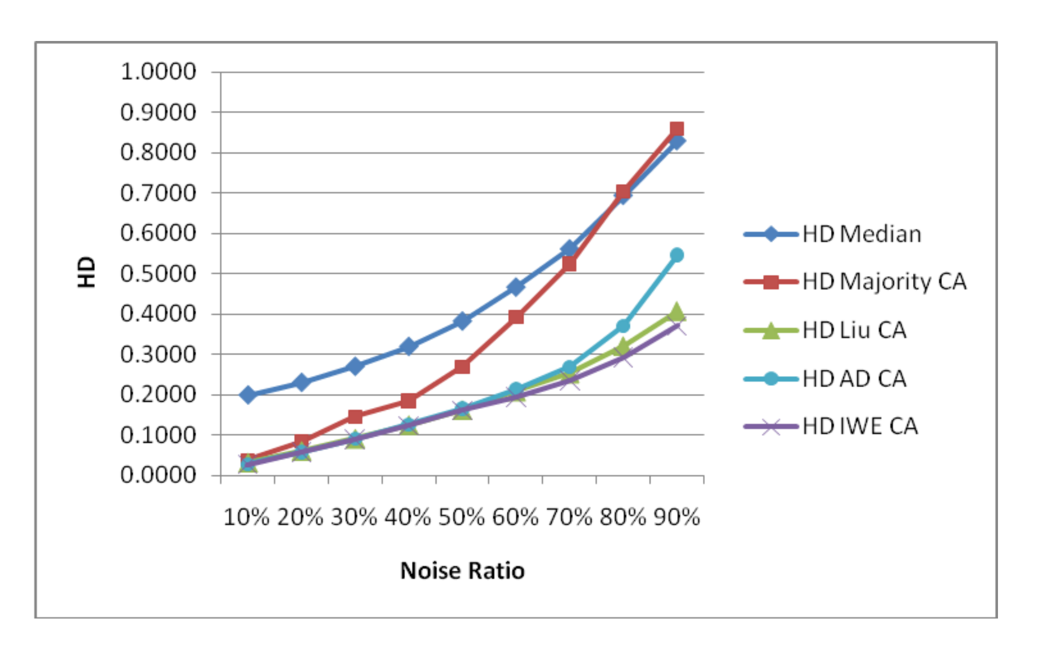


Figure (4.10) HD comparison results of applying different models on blown_ic_hr test image

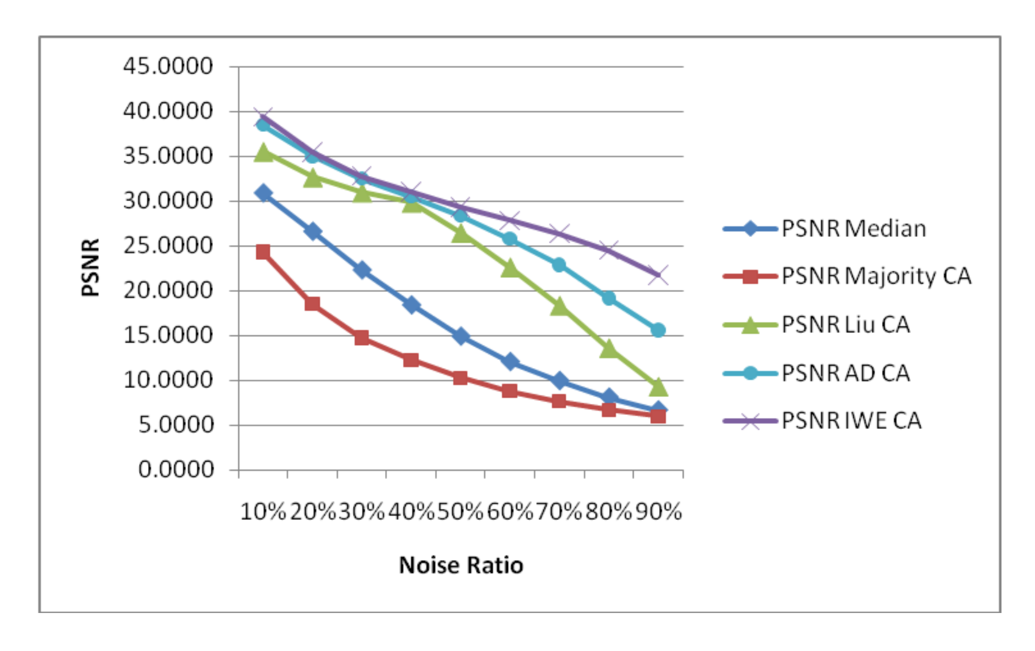


Figure (4.11) PSNR comparison results of applying different models on blown_ic_hr test image

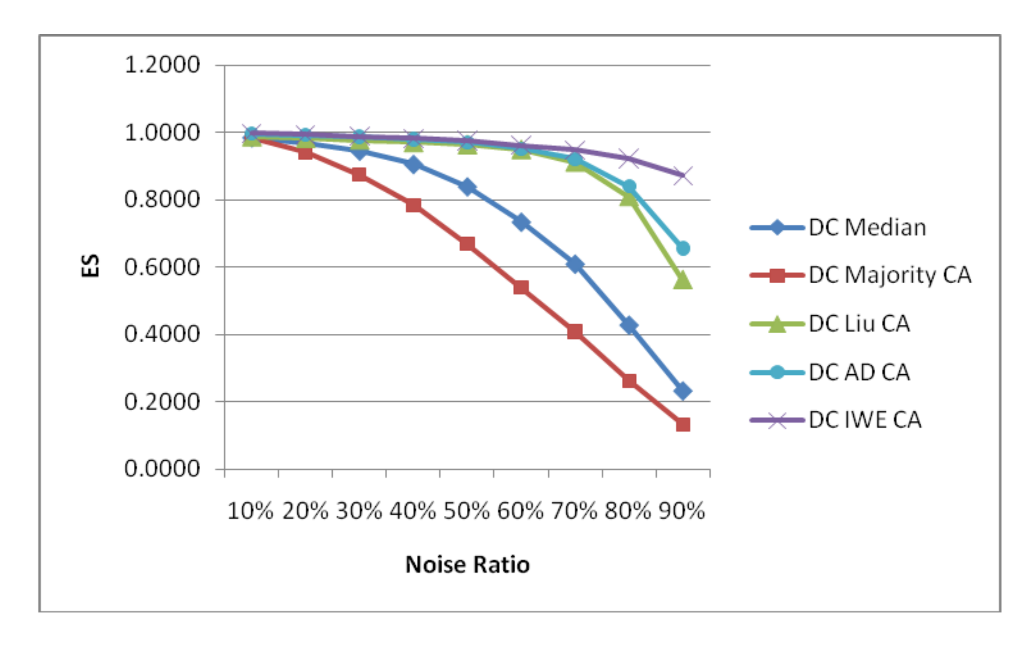


Figure (4.12) ES comparison results of applying different models on blown_ic_hr test image

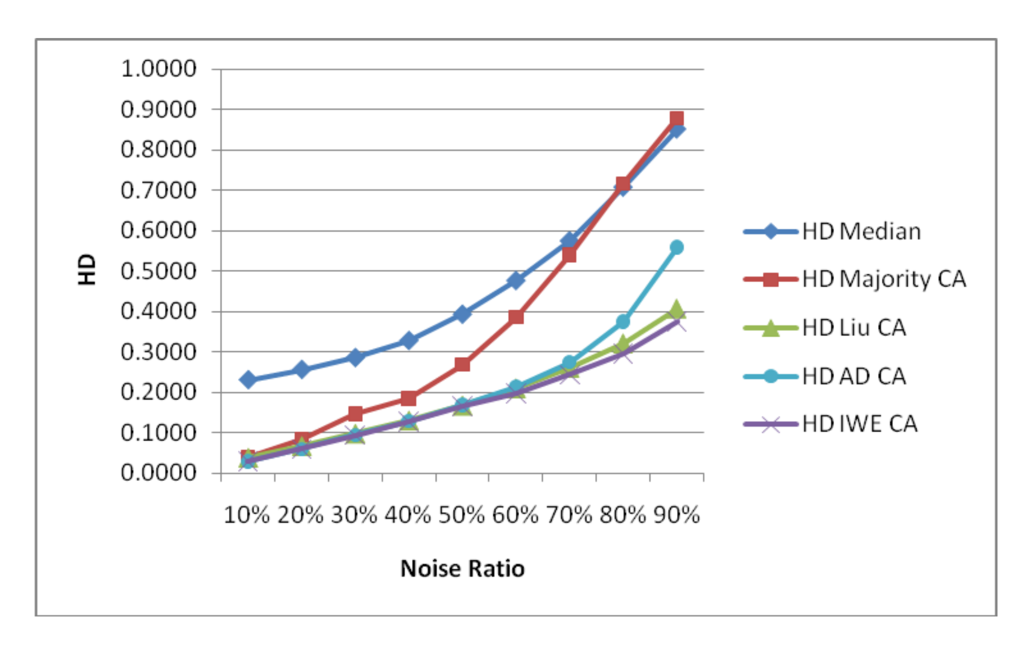


Figure (4.13) HD comparison results of applying different models on katrina_2005 test image

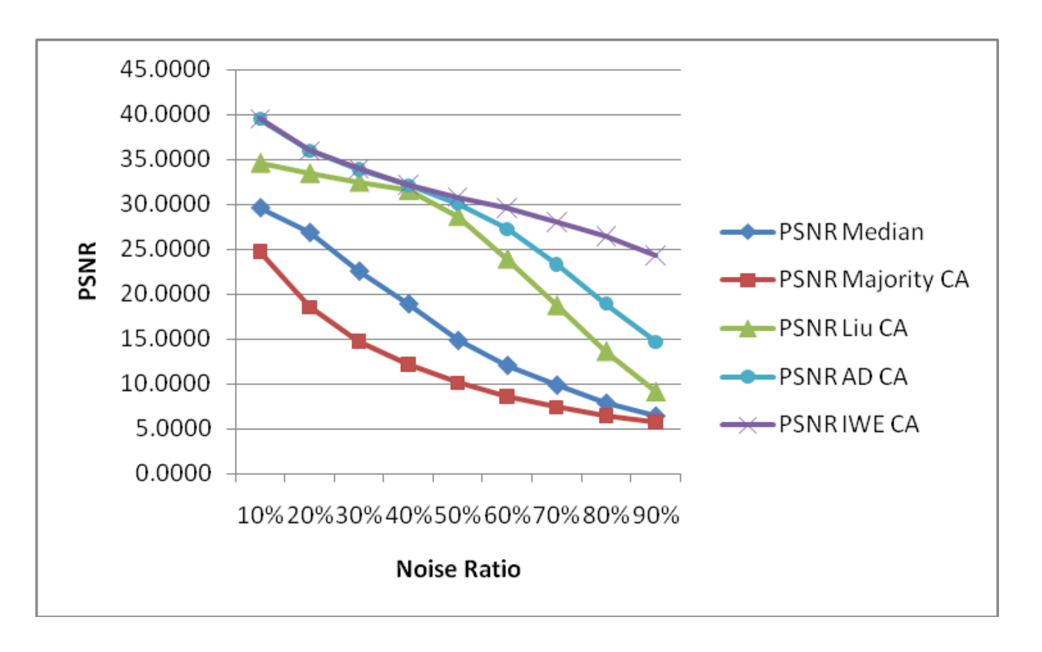


Figure (4.14) PSNR comparison results of applying different models on katrina_2005 test image

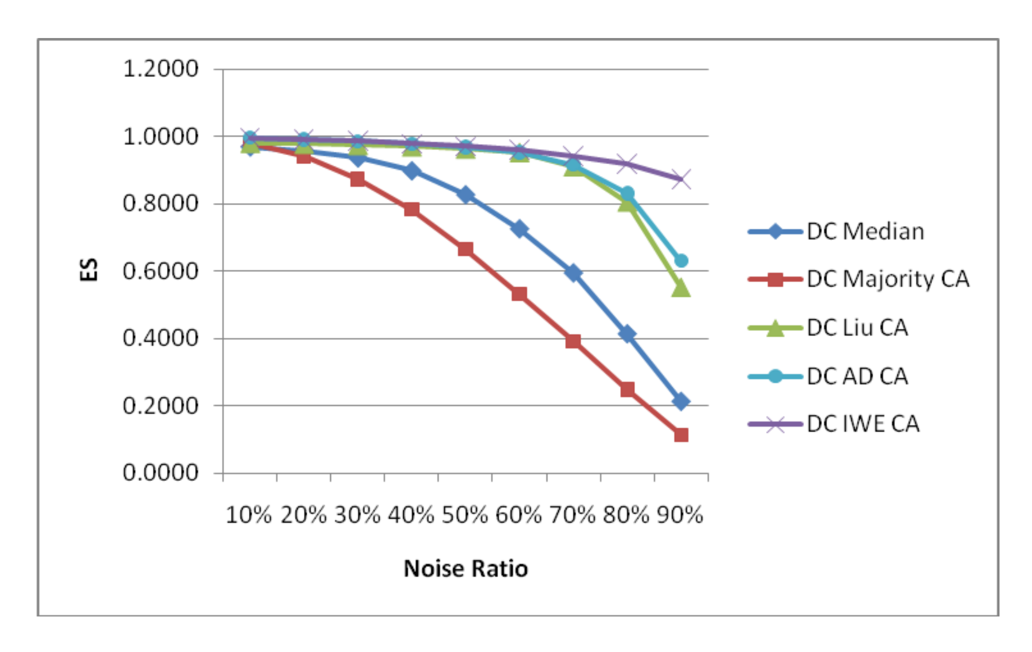


Figure (4.15) ES comparison results of applying different models on katrina_2005 test image

4.2 Experimental Results for Uniform Noise

The second set of experiments we executed on uniform noise, we executed experiments on our proposed IWE CA, (Selvapeter and Hordijk, 2009) “Majority” CA, and (Abu dalhoum, et al. 2011) CA on Lena, blown_ic_hr, Katrina_2005, and Skull test images, and then we compared the obtained results together.

Figure (4.16) shows results of applying IWE CA to restore Lena image noised with 10% and 40%, uniform noise.



Figure (4.16) Results of IWE CA on Lena test image
a: 10% uniform noised Lena image **b:** restored image of a using IWE CA
c: 40% uniform noised Lena image **d:** restored image of c using IWE CA

Figure (4.17) shows comparison between HD result values obtained by executing different algorithms on Lena test image containing uniform noise with different noise ratios.

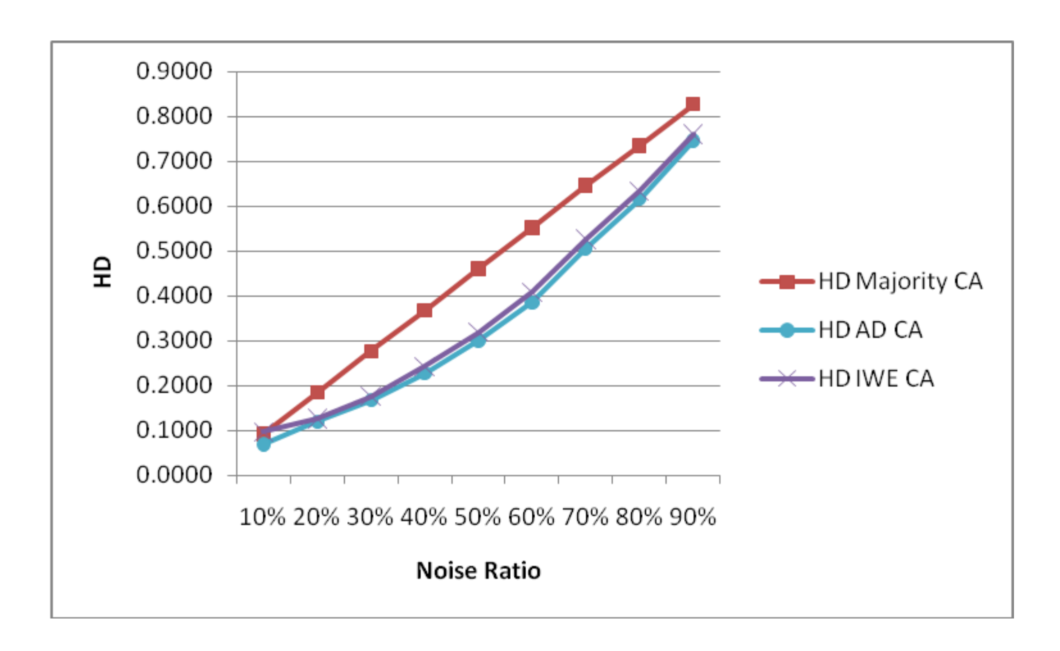


Figure (4.17) HD comparison results of applying different models on Lena test image

IWE CA gives HD values that are very close to (Abu dalhoum, et al. 2011) CA but not better than it.

Figure (4.18) shows comparison between PSNR result values obtained by applying different algorithms on different noise ratios of uniform noise added to Lena test image.

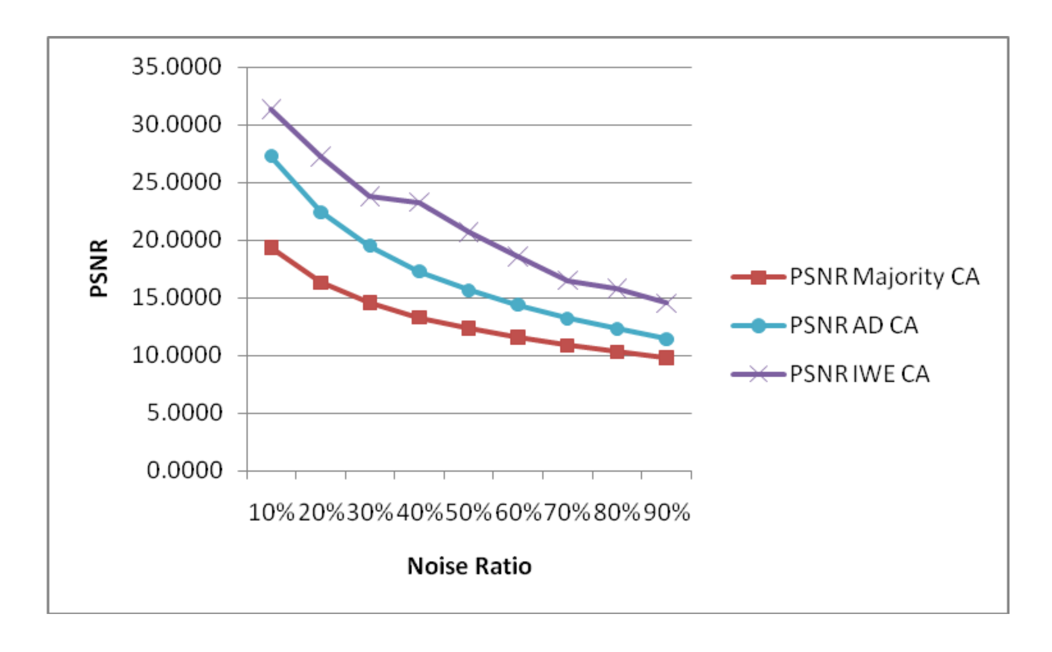


Figure (4.18) PSNR comparison results of applying different models on Lena test image

Our proposed IWE CA gives the maximum PSNR values over different noise ratios, and as the maximum value of PSNR means better performance; this means that IWE CA gives the best performance, then comes (Abu dalhoum, et al. 2011), then (Selvapeter and Hordijk, 2009) “Majority” CA.

Here we should note that PSNR is more accurate measurement than HD; because it gives the degree of similarity between two images, while HD gives only the number of different pixels between two images; so we consider PSNR values in Figure (4.18) to conclude that results of our proposed IWE CA outperforms result of (Abu dalhoum, et al. 2011) CA and (Selvapeter and Hordijk, 2009) “Majority” CA.

Figure (4.19) shows ES comparison between different algorithms applied on different uniform noise ratios added to Lena test image.

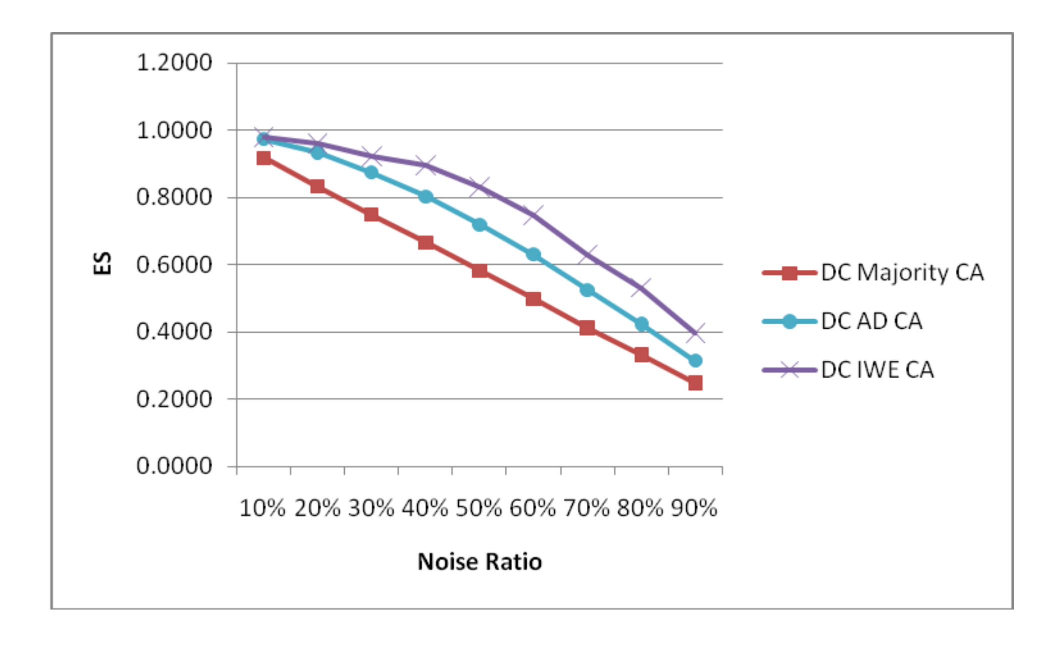


Figure (4.19) ES comparison results of applying different models on Lena test image

It is clear here that IWE CA gives maximum ES over other algorithms, and as the maximum value of ES means best performance; we conclude that our IWE CA outperforms other algorithms.

Figure (4.20) shows results of applying IWE CA to restore Wooden-boat image noised with 10% and 40%, uniform noise.

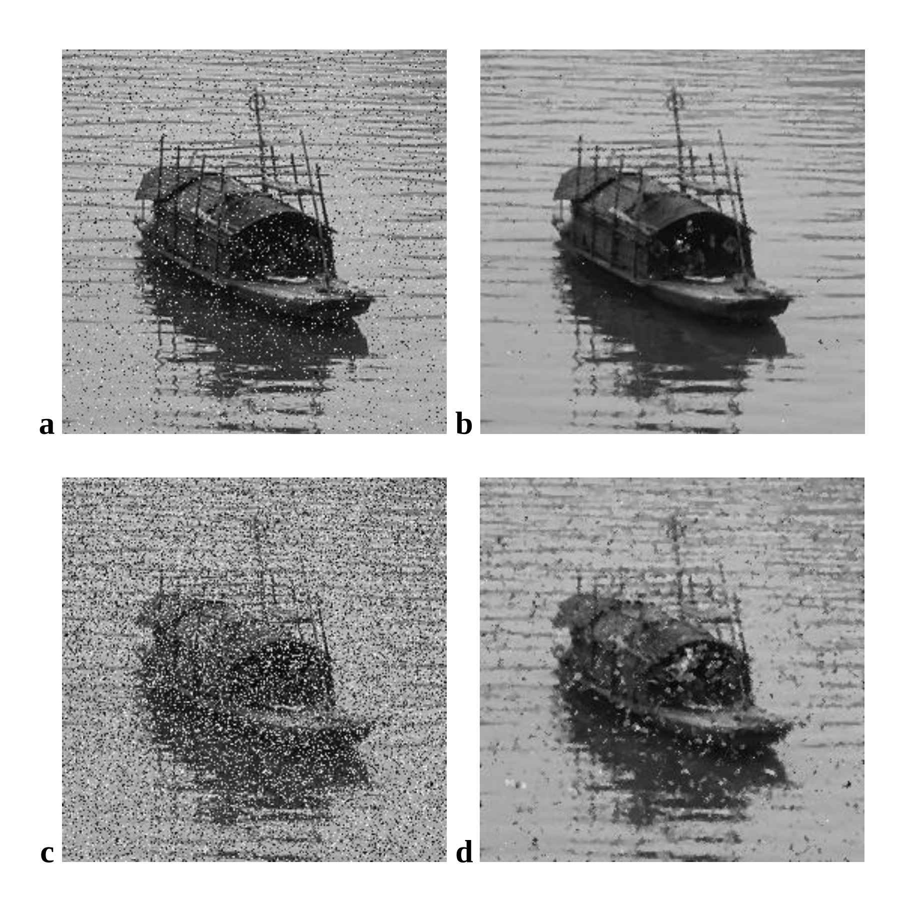


Figure (4.20) Results of IWE CA on Wooden-boat test image
a: 10% uniform noised Wooden-boat image **b:** restored image of a using IWE CA
c: 40% uniform noised Wooden-boat image **d:** restored image of c using IWE CA

Similarly, Figures (4.21) through (4.29) compare show that IWE CA outperforms (Abu dalhoum, et al. 2011) CA and (Selvapeter and Hordijk, 2009) “Majority” CA.

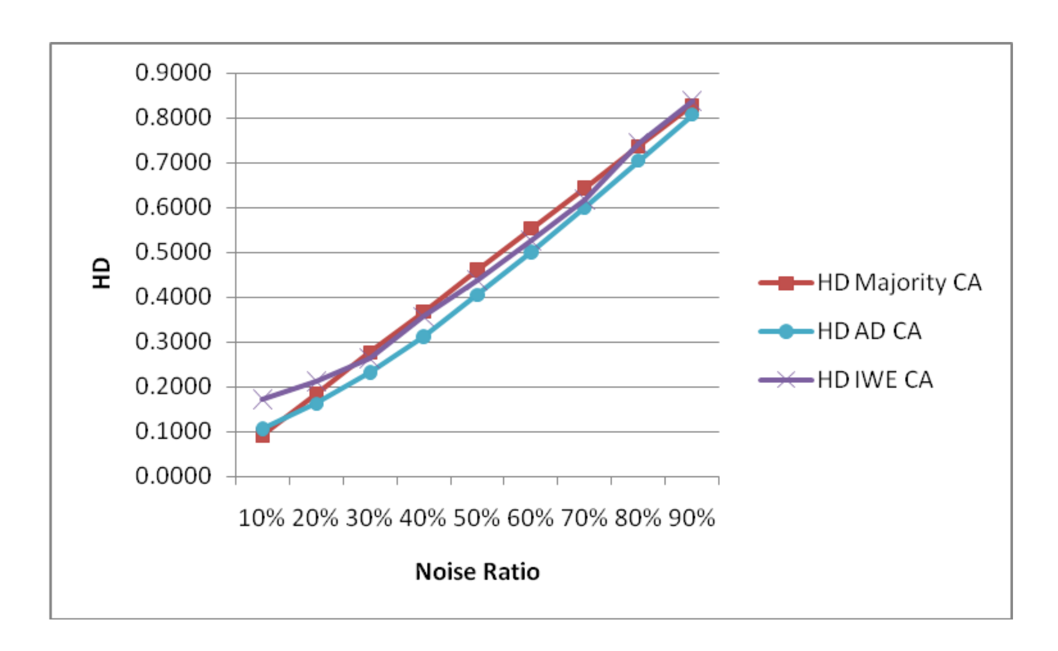


Figure (4.21) HD comparison results of applying different models on Wooden-boat test image

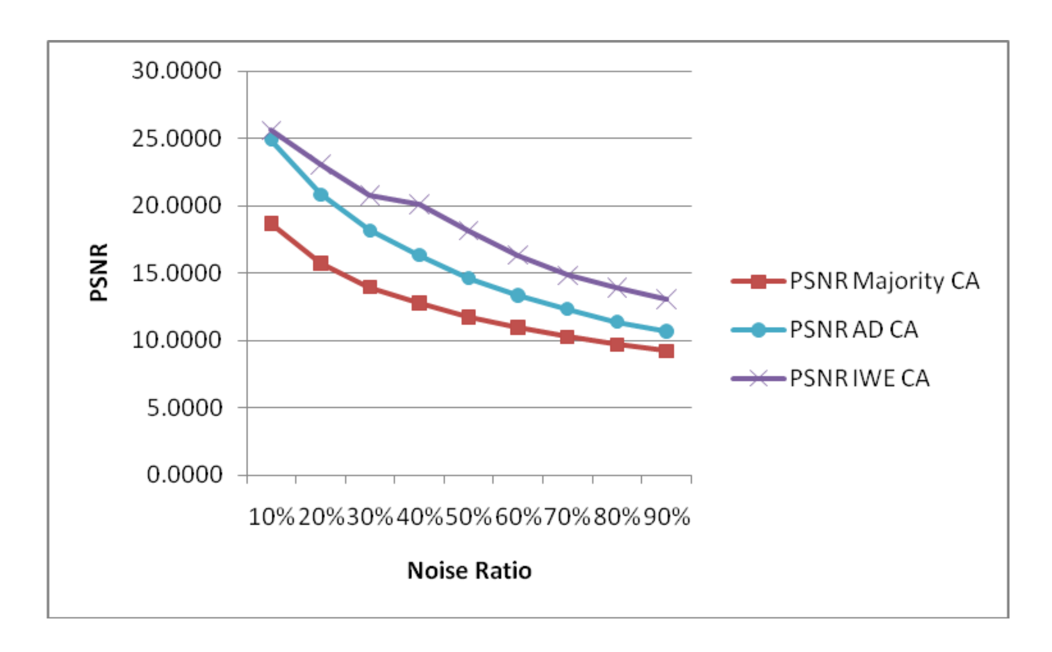


Figure (4.22) PSNR comparison results of applying different models on Wooden-boat test image

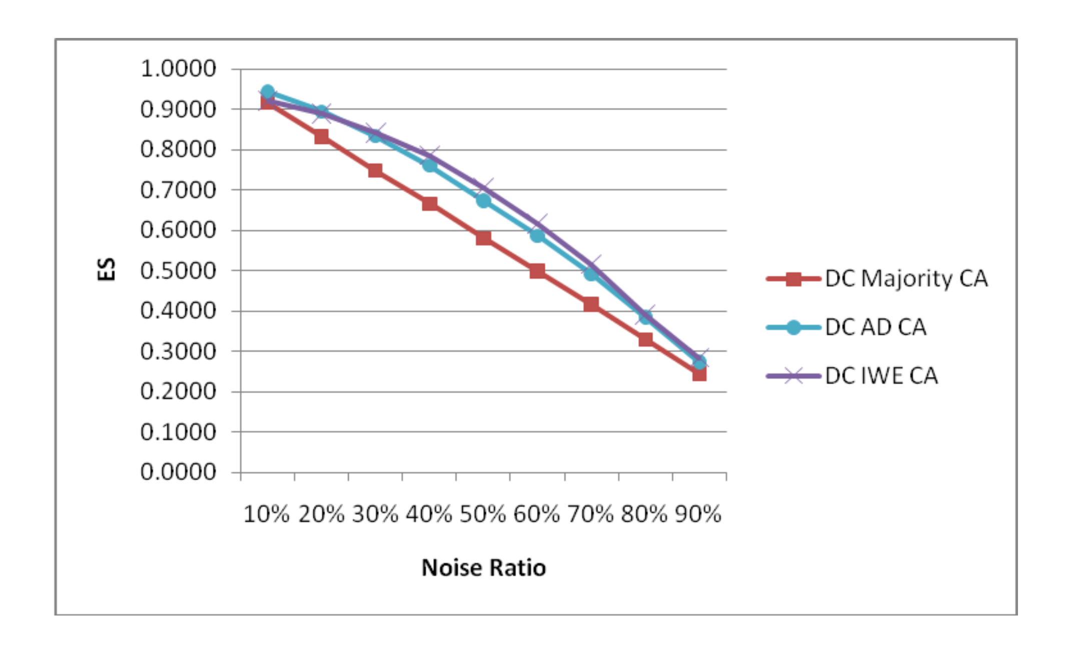


Figure (4.23) ES comparison results of applying different models on Wooden-boat test image

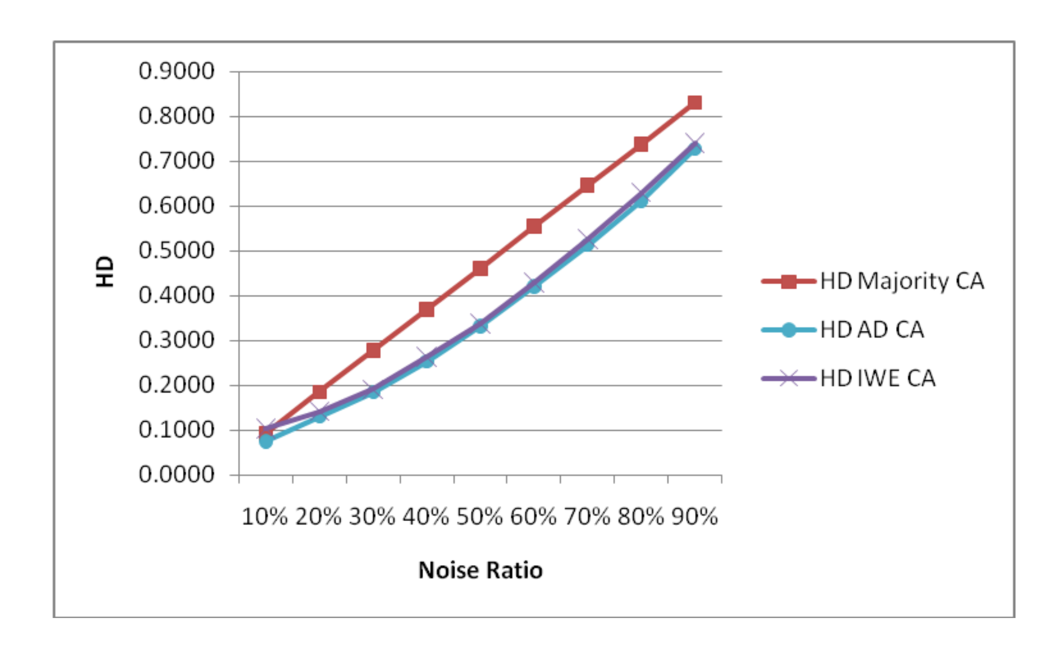


Figure (4.24) HD comparison results of applying different models on blown_ic_hr test image

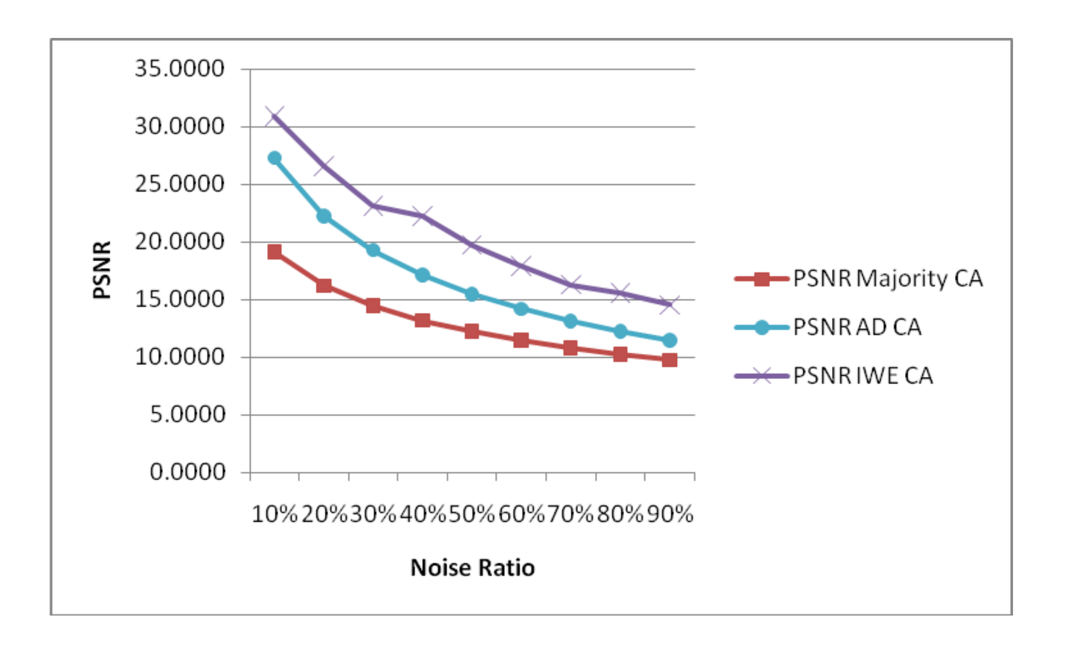


Figure (4.25) PSNR comparison results of applying different models on blown_ic_hr test image

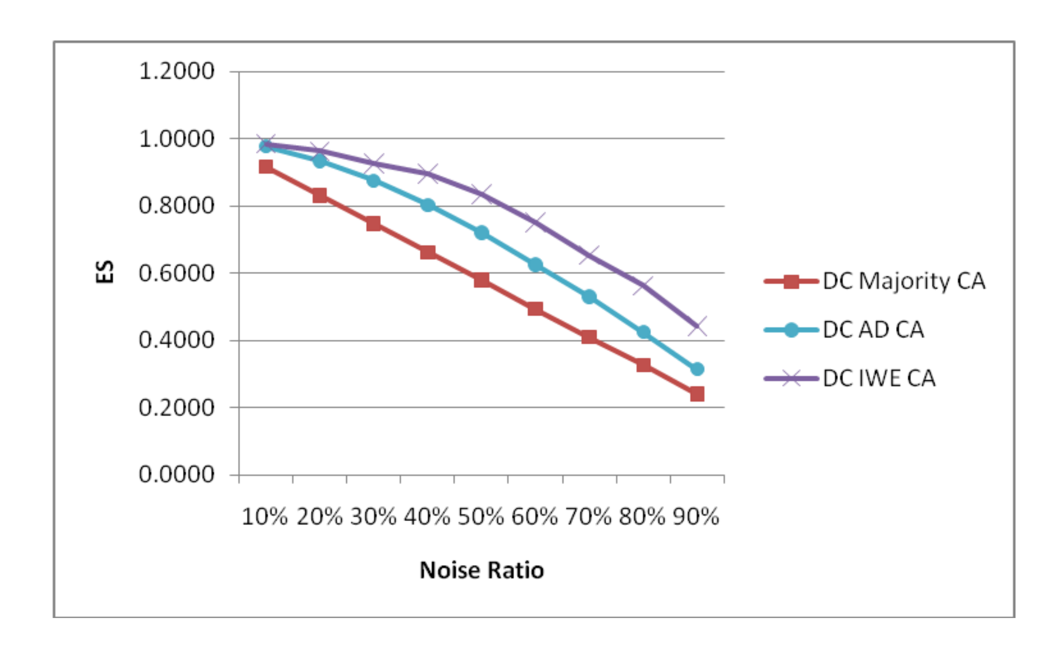


Figure (4.26) ES comparison results of applying different models on blown_ic_hr test image

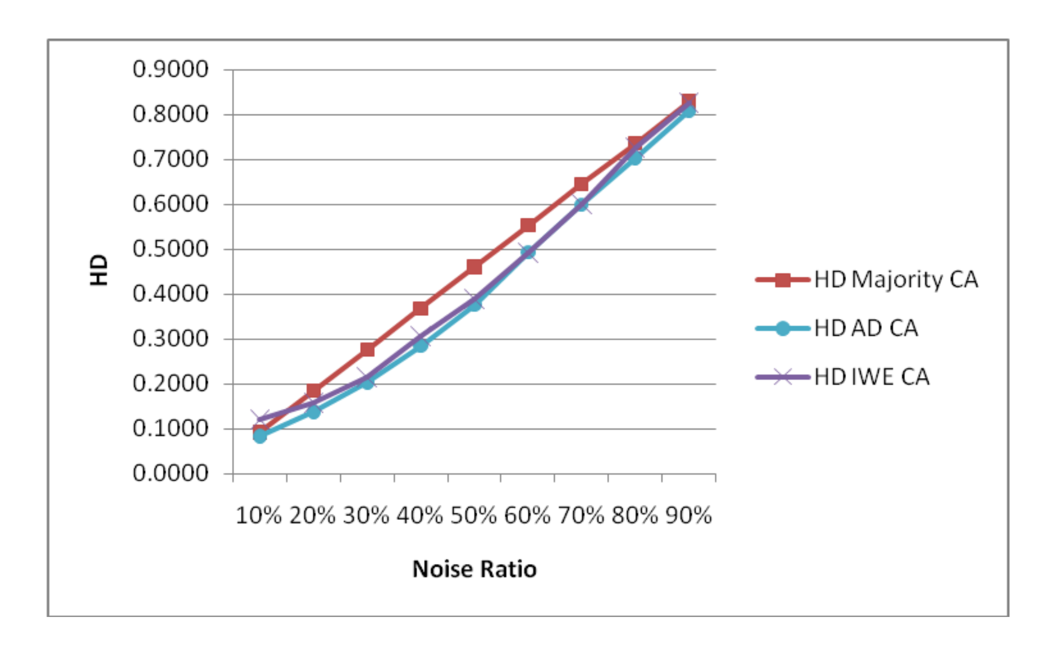


Figure (4.27) HD comparison results of applying different models on katrina_2005 test image

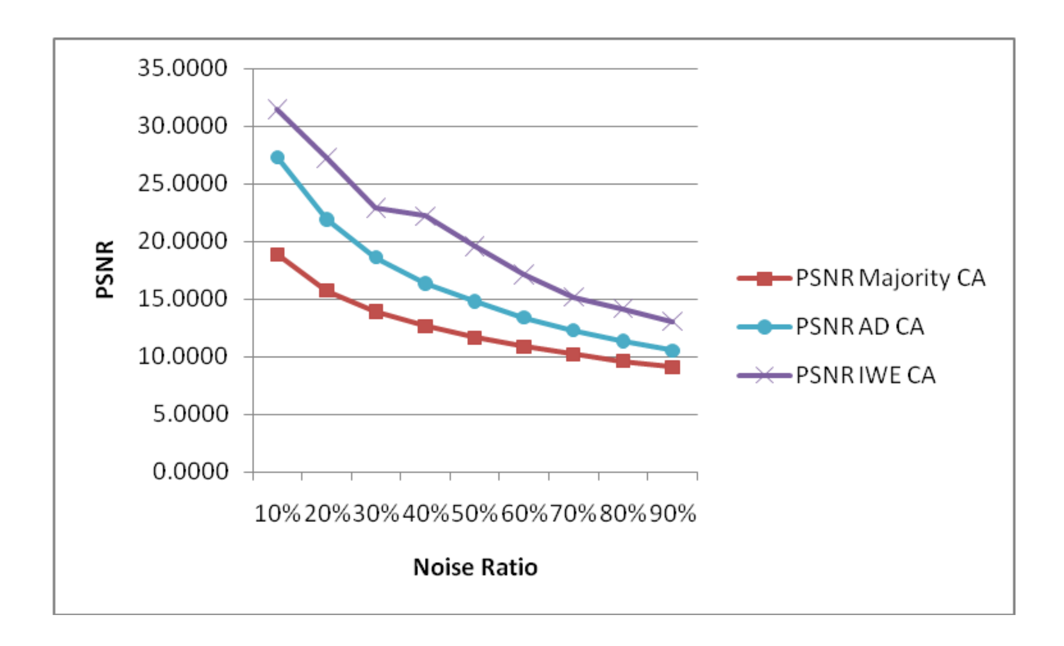


Figure (4.28) PSNR comparison results of applying different models on katrina_2005 test image

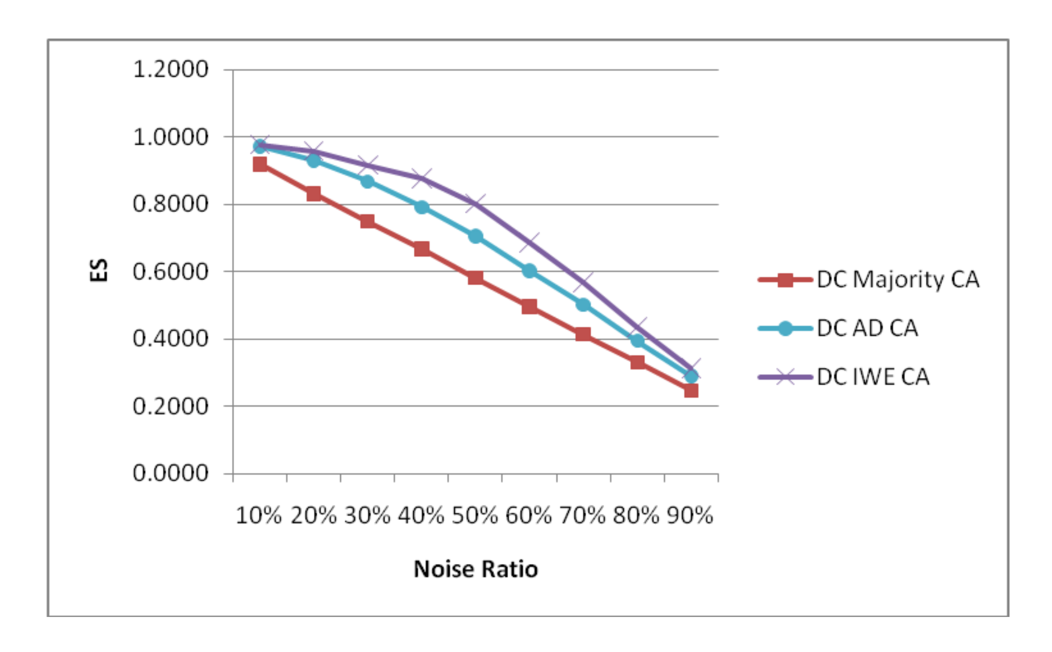


Figure (4.29) ES comparison results of applying different models on katrina_2005 test image

4.3 Conclusions

In this thesis, we studied (Popovici and Popovici, 2002), (Guieb, 2007), (Liu, et al. 2008), (Selvapeter and Hordijk, 2009), and (Abu dalhoum, et al. 2011) CA models, we tried to extract advantages and disadvantages of each of them in order to use this information in proposing a new CA enhanced model.

During our work, we were trying to introduce a specialized CA model for each noise type, and which is independent from the characteristics of the processed image, uses an improved technique to decrease the effect of noisy cells, and that supports iteration feature, and handles cells at the boundaries.

We proposed a new CA model for image noise removal, we called it Iterative Weighted Edges (IWE) CA model, it consists of three major steps of processing noised images: the first is Iterative CA, the second step gets the double edges of step 1 result

using Canny edge detector, the last step executes Iterative Weighted Edges CA.

Our IWE CA model works on gray scale images noised with salt & pepper or uniform noise, we applied it on different images of different fields; in order to approve that our model is dependent from the nature of the image and its type, and we have shown that our CA model outperforms other CA models studied in this paper, and it was obvious that it results in the most visually appealing denoised images.

4.4 Future Work

In future, our proposed CA can be studied on colored images, also a new CA model can be proposed to replace Canny edge detector to get double edges, future work can be done to merge all three steps of our model (Iterative CA, Canny double edge detection, and Weighted edges CA) in one unified step, and provide a measurement technique to support automatic decision to stop iteration process once no improvement is obtained if another iteration happens.

REFERENCES

Abdel latif Abu Dalhoum, Ibraheem al Dhamari, Alfonso Ortega, Manuel Alfonseca, **Enhanced Cellular Automata for Image Noise Removal**, Eurosis Publication , ASTEC'2011, March 1-3, 2011, vol.3 , pp.69-73

Boo, S.T. Ibrahim, H. and Vin Toh, K.K. (2009), An Improved Progressive Switching Median Filter, **International Conference on Future Computer and Communication, Kuala Lumpur**, Malaysia, 3-5 April, 2009.

Canny, J. (1986), **A Computational Approach to Edge Detection**, IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE, PAMI-8 (6).

Conway, J.H. (1976), **On Numbers And Games**, (2nd ed.), the United States of America: A K Peters, Ltd.

Efford, N. (2000), **Digital Image Processing: A Practical Introduction Using Java**, (1st ed.), United States of America: Pearson Education Limited.

Eng, H.L. and Ma, K.K. (2000), **Noise adaptive soft-switching median filter for image denoising**, IEEE International Conference on Acoustics, Speech, and Signal Processing, vol.4, 2175-2178.

Eng, H.L. and Ma, K.K. (2001), **Noise adaptive soft-switching median filter**, IEEE Transactions on Image Processing, 10 (2), 242 –251.

Ganguly, N. Sikdar, B. Deutsch, A. Canright, G. and Chaudhuri P. (2001), A Survey on Cellular Automata. Technical Report, **Centre for High Performance Computing**, Dresden University of Technology, Dresden, Germany.

Gauci, R. (2008), **A History of Cellular Automata**, University of East London, May 2008.

Ghanekar, A. Singh A.K. and Pandey R. (2007), A New Scheme for Impulse Detection in Switching Median Filters for Image Filtering, **International Conference on Computational Intelligence and Multimedia Applications 2007**, Sivakasi, India, 13-15 December, 2007.

Gobron, S. Finck, D. Even, P. and Kerautret B. (2006), **Merging Cellular Automata for Simulating Surface Effects**, Lecture Notes in Computer Science, 4173/2006, 94-103.

Gonzalez, R. and Woods, R. (2002), **Digital Image Processing**, (2nd ed.), United States of America: Prentice Hall.

Guieb, Eleni (2007), **Image Noise Reduction Using Cellular Automata**, Unpublished Bachelors Dissertation, University of the Philippines Los Ban˜os, Institute of Computer Science.

Jain, Anil (1989), **FUNDAMENTALS OF Digital IMAGE PROCESSING**, (1st ed.), United States of America, Prentice Hall.

Kulkarni R.K. Lahoti, C.B. and Meher S. (2010), Impulse Denoising Using Improved Progressive Switching Median Filter, **International Conference and Workshop on Emerging Trends in Technology (ICWET 2010)**, Mumbai, India, 26-27 February, 2010.

Kumar, T. and Sahoo, G. (2010), **A Novel Method of Edge Detection using Cellular Automata**, International Journal of Computer Applications (0975 – 8887), 9 (4).

Liu, S. Chen, H. and Yang, S. (2008), An Effective Filtering Algorithm for Image Salt-pepper Noises Based on Cellular Automata, **2008 Congress on Image and Signal Processing CISP**, Volume 3, Sanya, China, 27 – 30 May, 2008, 294 – 297.

Nadernejad, E. Sharifzadeh, S. and Hassanpour, H. (2008), **Applied Mathematical Sciences**, 2 (31), 1507 – 1520.

Neoh, H.S. and Hazanchuk, A. (2005), **Adaptive Edge Detection for Real-Time Video Processing using FPGAs**, Applications note, Altera corporation, 2005.

Niesche, Harald (2006), **Introduction to Cellular Automata**, Seminar “Organic Computing” SS2006.

Owens, R. (1997), **Computer Vision IT412, Lecture 6**, 10/29/1997, from **http://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL_COPIES/OWENS/LECT6/node2**

Popovici, A. and Popovici, D. (2002), **Cellular Automata in Image Processing, Unpublished Masters Dissertation**, University of the West Timisoara, Departments of Computer Science and Mathematics, Timisoara, Romania.

Rosin, P.L. (2010), **Image processing using 3-state cellular automata**, Computer Vision and Image Understanding 114(7), 790-802.

Selvapeter, P.J. and Hordijk P. (2009), **Cellular Automata for Image Noise Filtering,** In A. Abraham, A. Carvalho, F. Herrera and V. Pai (eds.), Proceedings of the World Congress on Nature and Biologically Inspired Computing 2009, 193–197.

Shapiro L.G. and Stockman G.C. (2001), **Computer Vision**, London, Prentice Hall.

Smith, S.M. and Brady, J.M. (1997), **SUSAN – A New Approach to Low Level Image Processing**, International Journal of Computer Vision, 23(1): 45-78.

Sutner, K. (1991), **Linear Cellular Automata and De Bruijn Automata**, Complex Systems, 5, 19–30

Thomas, C.D. (2000), **Evolution of Cellular Automata for Image Processing**, Unpublished Masters Dissertation, University of Birmingham, April 2000.

Wang, Z. and Zhang, D. (1999), **Progressive Switching Median Filter for the Removal of Impulse Noise from Highly Corrupted Images**, IEEE Trans. Circuits Syst., 46 (1), 78 – 80.

Wolfram, S. (1984), **Universality and Complexity in Cellular Automata**, Nonlinear Phenomena, 10 (1-2), 1-35.

Wolfram, S. (2002), **A New Kind of Science**, (1st ed.), United States of America, Wolfram Media.

Xiuqin, D. Yong, X. and Hong P. (2008), **A new kind of weighted median filtering algorithm used for image Processing**, 2008 International Symposium on Information Science and Engieering, 2, 738 - 743.

Ye, R. and Le, H. (2008), **A Novel Image Scrambling and Watermarking Scheme Based on Cellular Automata**, International Symposium on Electronic Commerce and Security, 3 (5), 938 – 941.

Young, Ian Theodore and Gerbrands, Jan Jacob and Van Vliet, Lucas Jozef (1998) **FUNDAMENTALS OF IMAGE PROCESSING**, (2nd ed.), Netherlands, Printed at the Delft University of Technology.

Zhang, S. Karim, M.A. (2002), **A new impulse detector for switching median filters**, IEEE Signal Processing Letters, 9 (11), 360-363.

Zhang, X. Yin, Z. and Xiong, Y. (2008), Adaptive Switching Mean Filter for Impulse Noise Removal, **2008 Congress on Image and Signal Processing CISP**, Volume 3, Sanya, China, 27 – 30 May, 2008, 275 – 278.

Zuse, K. (1969), **Calculating Space**, MIT Technical Translation AZT-70-164-GEMIT, MIT (Proj. MAC), Cambridge, Mass. 02139, Feb. 1970.

Appendix A: MATLAB Applications Source Code

(Liu, et al. 2008) Implementation

```
%======================================================================
%  Liu et al. (2008)  CA model
%======================================================================

function executer_Liu2008
clc;
originalImageName = 'Lena'; %bmp
%originalImageName = 'WoodenBoat'; %bmp
%originalImageName = 'Birds'; %bmp
%originalImageName = 'Cameraman'; %tif
%originalImageName = 'Rose'; %tif
% originalImageName = 'blown_ic_hr'; %tif
% originalImageName = 'katrina_2005'; %tif
% originalImageName = 'skull'; %tif

imageType = 'bmp';

noisePercent = 0.10;
iterations = 1;
threshold = 20;
hdThreshold = 20;

originalImage = imread(strcat('c:\CA work
area\',originalImageName,'.', imageType));
noiseImage1 = imnoise(originalImage,'salt & pepper',noisePercent);

noiseImage = mirroredEdges(noiseImage1);%mirrored.....

[imageHeight,imageWidth] = size(noiseImage);
resultImage = uint8(zeros(imageHeight,imageWidth));

%figure, imshow(originalImage)
figure, imshow(noiseImage)

for iteration=1:iterations
    for i=2:imageHeight-1
       for j=2:imageWidth-1
          resultImage(i,j) = Liu2008(noiseImage,i,j,threshold);
       end % j Loop
    end % i Loop
    noiseImage = resultImage;
end %iterations Loop

resultImage = resultImage([2:257],[2:257]); %mirrored.....
figure, imshow(resultImage);

%write result image to disk
imwrite(uint8(resultImage),strcat('c:\CA work
area\denoised_new\Liu2008\', originalImageName, '_',
int2str(uint8(100*noisePercent)), '_Matlab.', imageType),imageType);
```

```
%re-read noise image
noiseImage = noiseImage1;

%%%%%%%%%%Calculating HD and PSNR%%%%%%%%%%%%%%%%
%HD
HD_noisy = HD(noiseImage , originalImage, hdThreshold);
HD_CA = HD(resultImage , originalImage, hdThreshold);

%PSNR noisy,(Liu2008 CA), Median respectively
[SNR_noisy,MSE_noisy]=PSNR(double(noiseImage),double(originalImage));
[SNR_CA,MSE_CA]=PSNR(double(resultImage),double(originalImage));

%ES
ES_noisy = exactSimilarity(noiseImage , originalImage, threshold);
ES_CA = exactSimilarity(resultImage , originalImage, threshold);

file_1 = fopen('c:\CA work
area\denoised_new\Liu2008\results_Liu2008.txt','a');
fprintf(file_1,strcat(originalImageName, '_',
int2str(uint8(100*noisePercent)), '\n'));
fprintf(file_1,strcat('HD_noisy= ', num2str(HD_noisy), '  HD_CA= ',
num2str(HD_CA), '\n'));
fprintf(file_1,strcat('PSNR_noisy= ', num2str(SNR_noisy), '  PSNR_CA=
', num2str(SNR_CA), '\n'));
fprintf(file_1,strcat('ES_noisy= ', num2str(ES_noisy), '  ES_CA= ',
num2str(ES_CA), '\n'));
fprintf(file_1,'\n');

fclose(file_1);

end


function y=Liu2008(a,i,j,th)

    [h w d]=size(a);
    if i~=1 && j~=1 && i~=h && j~=w

      y1=a(i-1,j-1); y2=a(i-1, j); y3=a(i-1,j+1);
      y4=a(i  ,j-1); y5=a(i  , j); y6=a( i ,j+1);
      y7=a(i+1,j-1); y8=a(i+1, j); y9=a(i+1,j+1);
      ay=[y1,y2,y3,y4,y5,y6,y7,y8,y9];
      mx=max(ay);    mn=min(ay);

      if (y5>mn) && (y5<mx)
          %do nothing
          y=y5;
      elseif (mx==mn)|(mx==ay|mn==ay)    %step 2
           %step 3
           if mn~=0
              y=mn;
           elseif mx~=255
              y=mx;
           else
             y=y5;
           end
      else %step 4
         ym=~(mx==ay|mn==ay);
```

```
            m=mean(ay(ym));
            if abs(double(y5)-double(m))<th
               y=y5;
            else
               y=m;
            end
         end

     else
         y=a(i,j);
     end

end
```

(Selvapeter and Hordijk 2009) Implementation for Salt and Pepper Noise

```
%======================================================================
%  Selvapeter and Hordijk (2009) CA model for Salt and Pepper Noise
%  Mirrored Random Von Neumann
%======================================================================

function executer_SelvapeterAndHordijk2009
clc;
originalImageName = 'Lena'; %bmp

imageType = 'bmp';

noisePercent = 0.10;
iterations = 1;
threshold = 20;

originalImage = imread(strcat('c:\CA work
area\',originalImageName,'.', imageType));
noiseImage = imnoise(originalImage,'salt & pepper',noisePercent);

%figure, imshow(originalImage)
figure, imshow(noiseImage)

noiseImageMirrored = mirroredEdges(noiseImage);%mirrored.....
[imageHeightMirrored,imageWidthMirrored] = size(noiseImageMirrored);
resultImage = noiseImageMirrored;

for iteration=1:iterations

    for j=2:imageHeightMirrored-1
       for i=2:imageWidthMirrored-1

            startIndexI = i-1;
            endIndexI = i+1;
            startIndexJ = j-1;
            endIndexJ = j+1;

            MooreNeighbors =
noiseImageMirrored(startIndexJ:endIndexJ,startIndexI:endIndexI);
```

```
            VonNumannNeighbors(1) = MooreNeighbors(1,2);
            VonNumannNeighbors(2) = MooreNeighbors(2,1);
            VonNumannNeighbors(3) = MooreNeighbors(2,3);
            VonNumannNeighbors(4) = MooreNeighbors(3,2);

            if noiseImageMirrored(j,i) == 0 || noiseImageMirrored(j,i)
== 255

                r = zeros (1,256);
                for a = 1 : 4
                    r(double(VonNumannNeighbors(a))+1) =
r(double(VonNumannNeighbors(a))+1) + 1;
                end

                max = 0;
                maxIndex = 0;
                maxRepetitions = 1;
                for x=1 : 256
                    if r(x) > max
                        max = r(x);
                        maxIndex = x;
                        maxRepetitions = 1;
                    elseif r(x) == max
                        max = r(x);
                        maxRepetitions = maxRepetitions + 1;
                    end
                end

                whiteBlackCount = 0;
                if maxRepetitions == 1
                    resultImage(j,i) = maxIndex - 1;
                else
                    randIndex = ceil(rand(1)*4);
                    resultImage(j,i) = VonNumannNeighbors(randIndex);
                end

             end

        end % j Loop
    end % i Loop

    noiseImageMirrored = resultImage;

end %iterations Loop

resultImage = resultImage(2:imageHeightMirrored-
1,2:imageWidthMirrored-1);  %mirrored.....
figure, imshow(resultImage);

%write result image to disk
imwrite(uint8(resultImage),strcat('c:\CA work
area\denoised_new\Majority\Von-Neumann_Mirrored_Random_',
originalImageName, '_', int2str(uint8(100*noisePercent)), '_Matlab.',
imageType),imageType);

%%%%%%%%%Calculating HD, PSNR, ES%%%%%%%%%%%%%%%
%HD
HD_CA = HD(resultImage , originalImage, threshold);
```

```
%PSNR
[SNR,MSE]=PSNR(double(resultImage),double(originalImage));

%Exact Similarity
ES_CA = exactSimilarity(resultImage , originalImage, threshold);

file_1 = fopen('c:\CA work area\denoised_new\Majority\results_Von-
Neumann_Mirrored_Random_.txt','a');
fprintf(file_1,strcat('Von-Neumann_Mirrored_Random_',
originalImageName, '_', int2str(uint8(100*noisePercent)), ' PSNR= ',
num2str(SNR), '  HD= ', num2str(HD_CA), ' ES= ', num2str(ES_CA),
'\n'));
fclose(file_1);

end
```

(Selvapeter and Hordijk 2009) Implementation for Normal Noise

```
%=====================================================================
%  Selvapeter and Hordijk (2009) CA model for Normal Noise
%  Mirrored Random Von Neumann
%=====================================================================

function executer_SelvapeterAndHordijk2009_NN
clc;
originalImageName = 'Lena'; %bmp

imageType = 'bmp';

noisePercent = 0.10;
iterations = 1;
threshold = 20;

originalImage = imread(strcat('c:\CA work
area\',originalImageName,'.', imageType));
noiseImage = addNormalNoise(originalImage,noisePercent*100);

%figure, imshow(originalImage)
figure, imshow(noiseImage)

noiseImageMirrored = mirroredEdges(noiseImage);%mirrored.....
[imageHeightMirrored,imageWidthMirrored] = size(noiseImageMirrored);
resultImage = noiseImageMirrored;

for iteration=1:iterations

    for j=2:imageHeightMirrored-1
       for i=2:imageWidthMirrored-1

            startIndexI = i-1;
```

```
            endIndexI = i+1;
            startIndexJ = j-1;
            endIndexJ = j+1;

            MooreNeighbors =
noiseImageMirrored(startIndexJ:endIndexJ,startIndexI:endIndexI);

            VonNumannNeighbors(1) = MooreNeighbors(1,2);
            VonNumannNeighbors(2) = MooreNeighbors(2,1);
            VonNumannNeighbors(3) = MooreNeighbors(2,3);
            VonNumannNeighbors(4) = MooreNeighbors(3,2);

            if noiseImageMirrored(j,i) == 0 || noiseImageMirrored(j,i)
== 255

                r = zeros (1,256);
                for a = 1 : 4
                    r(double(VonNumannNeighbors(a))+1) =
r(double(VonNumannNeighbors(a))+1) + 1;
                end

                max = 0;
                maxIndex = 0;
                maxRepetitions = 1;
                for x=1 : 256
                    if r(x) > max
                        max = r(x);
                        maxIndex = x;
                        maxRepetitions = 1;
                    elseif r(x) == max
                        max = r(x);
                        maxRepetitions = maxRepetitions + 1;
                    end
                end

                whiteBlackCount = 0;
                if maxRepetitions == 1
                    resultImage(j,i) = maxIndex - 1;
                else
                    randIndex = ceil(rand(1)*4);
                    resultImage(j,i) = VonNumannNeighbors(randIndex);
                end

             end

         end % j Loop
    end % i Loop

    noiseImageMirrored = resultImage;

end %iterations Loop

resultImage = resultImage(2:imageHeightMirrored-
1,2:imageWidthMirrored-1);  %mirrored.....
figure, imshow(resultImage);

%write result image to disk
imwrite(uint8(resultImage),strcat('c:\CA work
area\denoised_new\Majority\Von-Neumann_Mirrored_Random_',
```

```
originalImageName, '_', int2str(uint8(100*noisePercent)), '_Matlab.',
imageType),imageType);

%%%%%%%%%%Calculating HD, PSNR, ES%%%%%%%%%%%%%%%%%%
%HD
HD_CA = HD(resultImage , originalImage, threshold);

%PSNR
[SNR,MSE]=PSNR(double(resultImage),double(originalImage));

%Exact Similarity
ES_CA = exactSimilarity(resultImage , originalImage, threshold);

file_1 = fopen('c:\CA work area\denoised_new\Majority\results_Von-
Neumann_Mirrored_Random_.txt','a');
fprintf(file_1,strcat('Von-Neumann_Mirrored_Random_',
originalImageName, '_', int2str(uint8(100*noisePercent)), ' PSNR= ',
num2str(SNR), '  HD= ', num2str(HD_CA), ' ES= ', num2str(ES_CA),
'\n'));
fclose(file_1);

end
```

(Abu dalhoum, et al. 2011) Implementation for Salt and Pepper Noise

```
%=======================================================================
%  Abu dalhoum 2011  CA model for Salt and Pepper Noise
%=======================================================================

function executer_Abudalhoum2011
clc;
originalImageName = 'Lena'; %bmp

imageType = 'bmp';

noisePercent = 0.10;
iterations = 1;
threshold = 20;
hdThreshold = 20;

originalImage = imread(strcat('c:\CA work
area\',originalImageName,'.', imageType));
noiseImage1 = imnoise(originalImage,'salt & pepper',noisePercent);

noiseImage = mirroredEdges(noiseImage1);%mirrored.....

[imageHeight,imageWidth] = size(noiseImage);
resultImage = uint8(zeros(imageHeight,imageWidth));

%figure, imshow(originalImage)
figure, imshow(noiseImage)
```

```
for iteration=1:iterations
    for i=2:imageHeight-1
       for j=2:imageWidth-1
          resultImage(i,j) = Abudalhoum2011(noiseImage,i,j,threshold);
       end % j Loop
    end % i Loop
    noiseImage = resultImage;
end %iterations Loop

resultImage = resultImage([2:257],[2:257]); %mirrored.....
figure, imshow(resultImage);

%write result image to disk
imwrite(uint8(resultImage),strcat('c:\CA work
area\denoised_new\Abudalhoum2011\', originalImageName, '_',
int2str(uint8(100*noisePercent)), '_Matlab.', imageType),imageType);

%%%%%%%%%%Calculating HD, PSNR, ES%%%%%%%%%%%%%%%
%HD
HD_CA = HD(resultImage , originalImage, hdThreshold);

%PSNR
[SNR_CA,MSE_CA]=PSNR(double(resultImage),double(originalImage));

%Exact Similarity
ES_CA = exactSimilarity(resultImage , originalImage, threshold);

file_1 = fopen('c:\CA work
area\denoised_new\Abudalhoum2011\results_Abudalhoum2011.txt','a');
fprintf(file_1,strcat(originalImageName, '_',
int2str(uint8(100*noisePercent)), ',', num2str(HD_CA),
',',num2str(SNR_CA), ',' , num2str(ES_CA)  , ','));
fprintf(file_1,strcat(num2str(iterations),' Iterations\n'));

fclose(file_1);

end



function y=Abudalhoum2011(a,i,j,th)

    [h w d]=size(a);
    if i~=1 && j~=1 && i~=h && j~=w

      y1=a(i-1,j-1); y2=a(i-1, j); y3=a(i-1,j+1);
      y4=a(i  ,j-1); y5=a(i  , j); y6=a( i ,j+1);
      y7=a(i+1,j-1); y8=a(i+1, j); y9=a(i+1,j+1);
      ay=[y1,y2,y3,y4,y5,y6,y7,y8,y9];

      if y5 == 0 || y5 == 255

          yfiltered=ay(~(0==ay|255==ay));
          if size(yfiltered,2) > 0
              y=median(yfiltered);
          else
              y=mean(ay);
```

```
            end

        else
            y=y5;
        end

    end

end
```

(Abu dalhoum, et al. 2011) Implementation for Normal Noise

```
%======================================================================
%  Abu dalhoum 2011  CA model for Normal Noise
%======================================================================

function executer_Abudalhoum2011_NN
clc;
originalImageName = 'Lena'; %bmp

imageType = 'bmp';

noisePercent = 0.10;
iterations = 1;
threshold = 20;
hdThreshold = 20;

originalImage = imread(strcat('c:\CA work
area\',originalImageName,'.', imageType));
noiseImage1 = addNormalNoise(originalImage,noisePercent*100);

noiseImage = mirroredEdges(noiseImage1);%mirrored.....

[imageHeight,imageWidth] = size(noiseImage);
resultImage = uint8(zeros(imageHeight,imageWidth));

%figure, imshow(originalImage)
figure, imshow(noiseImage)

for iteration=1:iterations
    for i=2:imageHeight-1
       for j=2:imageWidth-1
          resultImage(i,j) =
Abudalhoum2011_NN(noiseImage,i,j,threshold);
       end % j Loop
    end % i Loop
    noiseImage = resultImage;
end %iterations Loop

resultImage = resultImage([2:257],[2:257]); %mirrored.....
figure, imshow(resultImage);

%write result image to disk
imwrite(uint8(resultImage),strcat('c:\CA work
```

```
area\denoised_new\Abudalhoum2011\', originalImageName, '_',
int2str(uint8(100*noisePercent)), '_Matlab.', imageType),imageType);

%%%%%%%%%%Calculating HD, PSNR, ES%%%%%%%%%%%%%%%%
%HD
HD_CA = HD(resultImage , originalImage, hdThreshold);

%PSNR
[SNR_CA,MSE_CA]=PSNR(double(resultImage),double(originalImage));

%Exact Similarity
ES_CA = exactSimilarity(resultImage , originalImage, threshold);

file_1 = fopen('c:\CA work
area\denoised_new\Abudalhoum2011\results_Abudalhoum2011.txt','a');
fprintf(file_1,strcat(originalImageName, '_',
int2str(uint8(100*noisePercent)), ',', num2str(HD_CA),
',',num2str(SNR_CA), ',' , num2str(ES_CA) , ','));
fprintf(file_1,strcat(num2str(iterations),' Iterations\n'));

fclose(file_1);

end



function y=Abudalhoum2011_NN(a,i,j,th)

    [h w d]=size(a);
    if i~=1 && j~=1 && i~=h && j~=w

      y1=a(i-1,j-1); y2=a(i-1, j); y3=a(i-1,j+1);
      y4=a(i  ,j-1); y5=a(i  , j); y6=a( i ,j+1);
      y7=a(i+1,j-1); y8=a(i+1, j); y9=a(i+1,j+1);
      ay=[y1,y2,y3,y4,y5,y6,y7,y8,y9];

      mx=max(ay);   mn=min(ay);

      if y5 == mn || y5 == mx
          y=median(ay);
      else
          y=y5;
      end

    end

end
```


Iterative Weighted Edges for Salt and Pepper Noise

```
%======================================================================
%  Iterative Weighted Edges CA model for Salt and Pepper Noise
%======================================================================
```

```
function executer_IterativeWeightedEdgesCA
clc;
originalImageName = 'Lena'; %bmp

imageType = 'bmp';

iterations = 1;

noisePercent = 0.10;
threshold = 20;
hdThreshold = 20;

originalImage = imread(strcat('c:\CA work
area\',originalImageName,'.', imageType));
noiseImage1 = imnoise(originalImage,'salt & pepper',noisePercent);

noiseImage = mirroredEdges(noiseImage1);%mirrored.....

[imageHeight,imageWidth] = size(noiseImage);
resultImage = uint8(zeros(imageHeight,imageWidth));

%figure, imshow(originalImage)
figure, imshow(noiseImage)

for iteration=1:iterations
    for i=2:imageHeight-1
       for j=2:imageWidth-1
          resultImage(i,j) = IterativeCA(noiseImage,i,j,threshold);
       end % j Loop
    end % i Loop
    noiseImage = resultImage;
end %iterations Loop


[edgesImage,t] = edge(resultImage,'canny');
[edgesImage,t] = edge(edgesImage,'canny');

%figure, imshow(edgesImage)

noiseImage = mirroredEdges(noiseImage1);
for iteration=1:iterations
    for i=2:imageHeight-1
       for j=2:imageWidth-1
          resultImage(i,j) =
WeightedEdgesCA(noiseImage,edgesImage,i,j,threshold);
       end % j Loop
    end % i Loop
    noiseImage = resultImage;
end %iterations Loop

resultImage = resultImage([2:257],[2:257]); %mirrored.....
figure, imshow(resultImage);
imwrite(uint8(resultImage),strcat('c:\CA work area\denoised_new\IWE\',
originalImageName, '_', int2str(uint8(100*noisePercent)), '_Matlab.',
imageType),imageType);

%%%%%%%%%%Calculating HD, PSNR, and ES%%%%%%%%%%%%%%%%
```

```
%HD
HD_CA = HD(resultImage , originalImage, hdThreshold);

%PSNR
[SNR_CA,MSE_CA]=PSNR(double(resultImage),double(originalImage));

%%Exact Similarity
ES_CA = exactSimilarity(resultImage , originalImage, hdThreshold);

file_1 = fopen('c:\CA work
area\denoised_new\IWE\results_IWE.txt','a');
fprintf(file_1,strcat(originalImageName, '_',
int2str(uint8(100*noisePercent)), ',', num2str(HD_CA),
',',num2str(SNR_CA), ',' , num2str(ES_CA) , ','));
fprintf(file_1,strcat(num2str(iterations),' Iterations\n'));

fclose(file_1);

end



function y=IterativeCA(a,i,j,th)

    [h w d]=size(a);
    if i~=1 && j~=1 && i~=h && j~=w

      y1=a(i-1,j-1); y2=a(i-1, j); y3=a(i-1,j+1);
      y4=a(i  ,j-1); y5=a(i  , j); y6=a( i ,j+1);
      y7=a(i+1,j-1); y8=a(i+1, j); y9=a(i+1,j+1);
      ay=[y1,y2,y3,y4,y5,y6,y7,y8,y9];

      if y5 == 0 || y5 == 255

          yfiltered=ay(~(0==ay|255==ay));

          if size(yfiltered,2) > 0
              mx=max(yfiltered);   mn=min(yfiltered);
              if mx-mn <= 10
                  y=mean(yfiltered);
              else
                  y=median(yfiltered);
              end
          else
              y=y5;
          end

      else
          y=y5;
      end

    end

end
```

```
function y=WeightedEdgesCA(a,b,i,j,th)

    y=a(i,j);
    [h w d]=size(a);
    if i~=1 && j~=1 && i~=h && j~=w

      y1=a(i-1,j-1); y2=a(i-1, j); y3=a(i-1,j+1);
      y4=a(i  ,j-1); y5=a(i  , j); y6=a( i ,j+1);
      y7=a(i+1,j-1); y8=a(i+1, j); y9=a(i+1,j+1);
      ay=[y1,y2,y3,y4,y5,y6,y7,y8,y9];

      %b=logical(1.-b);
      b1=b(i-1,j-1); b2=b(i-1, j); b3=b(i-1,j+1);
      b4=b(i  ,j-1); b5=b(i  , j); b6=b( i ,j+1);
      b7=b(i+1,j-1); b8=b(i+1, j); b9=b(i+1,j+1);
      by=[b1,b2,b3,b4,b5,b6,b7,b8,b9];

      if (y5 == 0 || y5 == 255)

          if b5==1 && size(by(by==1),2)>2
              ay=[ay ay(logical(by))];
          end

          yfiltered=ay(~(0==ay|255==ay));
          mx=max(yfiltered);   mn=min(yfiltered);

          if size(yfiltered,2) > 0
              if mx-mn <= 10
                  y=mean(yfiltered);
              else
                  y=median(yfiltered);
              end
          else
              y=y5;
          end

      end

    end

end
```


Iterative Weighted Edges for Normal Noise

```
%======================================================================
%  Iterative Weighted Edges CA model for Normal Noise
%======================================================================

function executer_IterativeWeightedEdgesCA_NN
clc;
originalImageName = 'Lena'; %bmp
```

```
imageType = 'bmp';

iterations = 1;

noisePercent = 0.10;
threshold = 20;
hdThreshold = 20;

originalImage = imread(strcat('c:\CA work
area\',originalImageName,'.', imageType));
noiseImage1 = addNormalNoise(originalImage,noisePercent*100);

noiseImage = mirroredEdges(noiseImage1);%mirrored.....

[imageHeight,imageWidth] = size(noiseImage);
resultImage = uint8(zeros(imageHeight,imageWidth));

%figure, imshow(originalImage)
figure, imshow(noiseImage)

for iteration=1:iterations
    for i=2:imageHeight-1
       for j=2:imageWidth-1
          resultImage(i,j) = IterativeCA_NN
(noiseImage,i,j,threshold);
       end % j Loop
    end % i Loop
    noiseImage = resultImage;
end %iterations Loop


[edgesImage,t] = edge(resultImage,'canny');
[edgesImage,t] = edge(edgesImage,'canny');

%figure, imshow(edgesImage)

noiseImage = mirroredEdges(noiseImage1);
for iteration=1:iterations
    for i=2:imageHeight-1
       for j=2:imageWidth-1
          resultImage(i,j) = WeightedEdgesCA_NN
(noiseImage,edgesImage,i,j,threshold);
       end % j Loop
    end % i Loop
    noiseImage = resultImage;
end %iterations Loop

resultImage = resultImage([2:257],[2:257]); %mirrored.....
figure, imshow(resultImage);
imwrite(uint8(resultImage),strcat('c:\CA work area\denoised_new\IWE\',
originalImageName, '_', int2str(uint8(100*noisePercent)), '_Matlab.',
imageType),imageType);

%%%%%%%%%%Calculating HD, PSNR, and ES%%%%%%%%%%%%%%%
%HD
HD_CA = HD(resultImage , originalImage, hdThreshold);

%PSNR
```

```
[SNR_CA,MSE_CA]=PSNR(double(resultImage),double(originalImage));

%%Exact Similarity
ES_CA = exactSimilarity(resultImage , originalImage, hdThreshold);

file_1 = fopen('c:\CA work
area\denoised_new\IWE\results_IWE.txt','a');
fprintf(file_1,strcat(originalImageName, '_',
int2str(uint8(100*noisePercent)), ',', num2str(HD_CA),
',',num2str(SNR_CA), ',' , num2str(ES_CA) , ','));
fprintf(file_1,strcat(num2str(iterations),' Iterations\n'));

fclose(file_1);

end



function y=IterativeCA_NN(a,i,j,th)

    [h w d]=size(a);
    if i~=1 && j~=1 && i~=h && j~=w

      y1=a(i-1,j-1); y2=a(i-1, j); y3=a(i-1,j+1);
      y4=a(i  ,j-1); y5=a(i  , j); y6=a( i ,j+1);
      y7=a(i+1,j-1); y8=a(i+1, j); y9=a(i+1,j+1);
      ay=[y1,y2,y3,y4,y5,y6,y7,y8,y9];

      mx=max(ay);   mn=min(ay);
      if y5 == mn || y5 == mx

          yfiltered=ay(~(mn==ay|mx==ay));

          if size(yfiltered,2) > 0
              if mx-mn <= 10
                  y=mean(yfiltered);
              else
                  y=median(yfiltered);
              end
          else
              y=y5;
          end

      else
          y=y5;
      end

    end

end



function y=WeightedEdgesCA_NN(a,b,i,j,th)

    y=a(i,j);
```

```
    [h w d]=size(a);
    if i~=1 && j~=1 && i~=h && j~=w

      y1=a(i-1,j-1); y2=a(i-1, j); y3=a(i-1,j+1);
      y4=a(i  ,j-1); y5=a(i  , j); y6=a( i ,j+1);
      y7=a(i+1,j-1); y8=a(i+1, j); y9=a(i+1,j+1);
      ay=[y1,y2,y3,y4,y5,y6,y7,y8,y9];

      %b=logical(1.-b);
      b1=b(i-1,j-1); b2=b(i-1, j); b3=b(i-1,j+1);
      b4=b(i  ,j-1); b5=b(i  , j); b6=b( i ,j+1);
      b7=b(i+1,j-1); b8=b(i+1, j); b9=b(i+1,j+1);
      by=[b1,b2,b3,b4,b5,b6,b7,b8,b9];

      mx=max(ay);   mn=min(ay);
      if y5 == mn || y5 == mx

          if b5==1  && size(by(by==1),2)>2
              ay=[ay ay(logical(by))];
          end

          yfiltered=ay(~(mn==ay|mx==ay));

          if size(yfiltered,2) > 0
              if mx-mn <= 10
                  y=mean(yfiltered);
              else
                  y=median(yfiltered);
              end
          else
              y=y5;
          end

      end

    end

end
```

Measurement Functions

```
%=====================================================================
%   HD Function
%=====================================================================

function t=HD(a,b,th)
    %convert to one dim
    na=a(:); nb=b(:);
    l=length(na);
    d=bitxor(na,nb);
    c=size(find(d >= th),1);
    t=c/l;
end
```

```
%=====================================================================
%   PSNR Function
%=====================================================================

function [decibels,MSE] = PSNR(A,B)

    if A == B
       error('Images are identical: PSNR has infinite value')
    end

    max2_A = max(max(A));
    max2_B = max(max(B));
    min2_A = min(min(A));
    min2_B = min(min(B));

    if max2_A > 255 || max2_B > 255 || min2_A < 0 || min2_B < 0
      error('input matrices must have values in the interval [0,255]')
    end

    error_diff = A - B;
    MSE = mean(mean(error_diff.^2));
    decibels = 20*log10(255/(sqrt(MSE)));
    disp(sprintf('PSNR = +%5.2f dB',decibels))

end



%=====================================================================
%   ES Function
%=====================================================================

function t=exactSimilarity(a,b,th)
    %convert to one dim
    na=a(:); nb=b(:);
    l=length(na);
    d=abs(double(na)-double(nb));
    c=size(find(d <= th),1);
    t=c/l;
end
```

Other Helping Functions

```
%=====================================================================
%   Mirrored Edges
%=====================================================================


function NA=addNormalNoise(A,sz)
% add uniform noise to an image with sz% of the image
% convert to two dimensions
clc;
```

```
if sz>100
   disp ('noise size ratio must be 0 to 100') ;
   NA=0;
   return
end
 [h w d]=size(A);
 if d==3
    A=rgb2gray(A);
 end
[h w d]=size(A);
s=h*w*d;
nsz=sz*s/100;

%choose pixels random pixels
r=1:s;
%linear random unique integers
for i=1:s
    rn=round((s-1)*rand(1,1))+1;
    tmp=r(i);
    r(i)=r(rn);
    r(rn)=tmp;
end

%fill with random unifrom noise
%map 1D to 2D row wise
for i=1:nsz
    xy=r(i);
    y=mod(xy,w);
    if y==0
       y=w;
       x=floor(xy/w);
    else
       x=floor(xy/w)+1;
    end
   A(x,y)=round(255*rand(1,1));
end
NA=A;
clear A r



%======================================================================
%   Mirrored Edges
%======================================================================


function y=mirroredEdges(a)

    [imageHeight,imageWidth] = size(a);

    y = uint8(zeros(imageHeight+2,imageWidth+2));

    y([2:257],[2:257]) = a;

    y(1,[2:257]) = a(1,[1:256]);
    y([2:257],1) = a([1:256],1);
    y(258,[2:257]) = a(256,[1:256]);
    y([2:257],258) = a([1:256],256);

    y(1,1) = a(1,1);
```

```
    y(1,258) = a(1,256);
    y(258,1) = a(256,1);
    y(258,258) = a(256,256);

end


%======================================================================
%  Mirrored Edges for Logical Image
%======================================================================


function y=mirroredEdgesLogical(a)

    [imageHeight,imageWidth] = size(a);

    y = logical(zeros(imageHeight+2,imageWidth+2));
    y([2:257],[2:257]) = a;

    y(1,[2:257]) = a(1,[1:256]);
    y([2:257],1) = a([1:256],1);
    y(258,[2:257]) = a(256,[1:256]);
    y([2:257],258) = a([1:256],256);

    y(1,1) = a(1,1);
    y(1,258) = a(1,256);
    y(258,1) = a(256,1);
    y(258,258) = a(256,256);

end
```

تحسين الصور الرقمية باستخدام الخلايا الآلية

اعداد
احمد رمضان حمدان

المشرف
الدكتور عبداللطيف ابو دلهوم

المشرف المشارك
الدكتور محمد قطاونة

ملخـــــــــــص

إن ازدياد أعداد البرمجيات التي تستخدم معالجة الصور الرقمية في مختلف الحقــول يزيــد الحاجة الى تطوير وتحسين تقنيات وأدوات معالجة الصور. العديد من القرارات الحرجــة يــتم اتخاذها على ضوء نتائج عمليات معالجة الصور، وهذه القرارات في بعض الأحيان قــد تــؤثر على حياة الإنسان كما في عمليات معالجة الصور المستخدمة في البرامج والأجهزة الطبية.

الخلايا الآلية هي عبارة عن نماذج حسابية لا مركزية، وهي تقنية فعالة وبسيطة وســريعة لحل المشكلات ونمذجة بعض العمليات، بالرغم من بساطتها، فإنها قادرة على القيــام بعمليــات حسابية معقدة بالإعتماد على المعلومات المحلية فقط، هذه الخصائص تجعل تطبيقها فــي مجــال معالجة الصور ناجحا.

العديد من الدراسات قامت بتوظيف الخلايا الآلية في معالجة الصور، فــي هــذه الرســالة سنقوم بطرح نموذج جديد ومطور لتحسين ومعالجة الصور باستخدام الخلايا الآلية. في البدايــة سنقوم بدراسة وتحليل وتطبيق النماذج الموجودة حاليا برمجيا وســنتعرف علــى نقــاط قوتهــا وضعفها، ثم سنطرح نموذجنا الجديد وسنقوم بتطبيقه برمجيا، وأخيرا سنقارن نتــائج نموذجنــا المقترح مع نتائج النماذج الحالية.